A Framework for the Design, Prototyping and Evaluation of Mobile Interfaces for Domestic Environments

Patrick Michael Holroyd

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy at the
University of Sussex

School of Engineering and Informatics

University of Sussex

Brighton

BN1 9QJ

September 2012

Declaration

I hereby declare that this thesis has not been and will not be, submitted in whole or in part to another University for the award of any other degree.

Signed……

Patrick Michael Holroyd

Acknowledgements

This thesis would have not have been possible without the support of my supervisors: Dr Phil Watten and Dr Paul Newbury. I would like to thank them for their help, encouragement and guidance throughout this lengthy process.

I would also like to thank Natasha and my family for their support and unfaltering confidence in me.

University of Sussex

Patrick Michael Holroyd

Submitted for the degree of Doctor of Philosophy

A Framework for the Design, Prototyping and Evaluation of Mobile Interfaces for Domestic Environments

Abstract

The idea of the smart home has been discussed for over three decades, but it has yet to achieve mass-market adoption. This thesis asks the question *Why is my home not smart?* It highlights four main areas that are barriers to adoption, and concentrates on a single one of these issues: usability. It presents an investigation that focuses on design, prototyping and evaluation of mobile interfaces for domestic environments resulting in the development of a novel framework.

A smart home is the physical realisation of a ubiquitous computing system for domestic living. The research area offers numerous benefits to end-users such as convenience, assistive living, energy saving and improved security and safety. However, these benefits have yet to become accessible due to a lack of usable smart home control interfaces. This issue is considered a key reason for lack of adoption and is the focus for this thesis.

Within this thesis, a framework is introduced as a novel approach for the design, prototyping and evaluation of mobile interfaces for domestic environments. Included within this framework are three components. Firstly, the Reconfigurable Multimedia Environment (RME), a physical evaluation and observation space for conducting user centred research. Secondly, Simulated Interactive Devices (SID), a video-based development and control tool for simulating interactive devices commonly found within a smart home. Thirdly, iProto, a tool that facilitates the production and rapid deployment of high fidelity prototypes for mobile touch screen devices. This framework is evaluated as a round-tripping toolchain for prototyping smart home control and found to be an efficient process for facilitating the design and evaluation of such interfaces.

List of Publications

Holroyd, P., P. Watten, and P. Newbury, Why is my home not smart?, Aging Friendly Technology for Health and Independence, 8th International Conference on Smart Homes and Health Telematics, ICOST 2010, Seoul, Korea, June 22-24, 2010. Proceedings. Lecture Notes in Computer Science 6159 Springer 2010, ISBN 978-3-642-13777-8. p. 53-59.

Holroyd, P., P. Watten, and P. Newbury, Reconfigurable Multimedia Environment. Journal of Emerging Technologies in Web Intelligence, 2011. 3(5), ISSN 1798-046. p. 282-285.

Holroyd, P., P. Watten, and P. Newbury, Reconfigurable Multimedia Environment, in Proceedings of the 5th international conference on Ubiquitous and Collaborative Computing September 2010, British Computer Society: Abertay, Dundee. p. 6-10.

Rivera, F., P. Watten, **P. Holroyd**, F. Beacher, K. Mania, H. Critchley, Real-time compositing framework for interactive stereo fMRI displays, in ACM SIGGRAPH 2010 Posters 2010, ACM: Los Angeles, California. p. 1-1.

Newbury, P., P. Watten, **P. Holroyd**, C. Hardman, Why Recording Lectures Requires a New Approach, in 10th European Conference on eLearning. December 2011. Brighton: Academic Conferences International.

Glossary

API	Application Programming Interface
APP	Mobile Application
BCI	Brain-Computer Interface
CAVE	Cave Automatic Virtual Environment
CDM	Core Data Model
CG	Computer-Generated
EFM	Extensible Feedback Mechanism
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HD	High Definition
HMD	Head Mounted Display
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
IoT	Internet of Things
IR	Infrared
LAN	Local Area Network
MOC	Managed Object Context
MIDI	Musical Instrument Digital Interface
NAHB	National Association of Home Builders
NLP	Natural Language Processing
OEM	Original Equipment Manufacturer
OpenGL	Open Graphics Library
OS	Operating System
OU	Open University
PAL	Phase Alternating Line
PDA	Personal Digital Assistant
PNG	Portable Network Graphic
PS	Photoshop
R&D	Research and Development
RFID	Radio-frequency Identification

RME	Reconfigurable Multimedia Environment
RS-232	Recommended Standard 232
SDK	Software Development Kit
SID	Simulated Interactive Devices
UI	User Interface
UID	Unique Identifier
UML	Unified Modelling Language
URC	Universal Remote Control
UX	User Experience
UXD	User Experience Design
WLAN	Wireless Local Area Network
WOz	Wizard of Oz
XML	Extensible Mark-up Language

Contents

Figures

Tables

Chapter 1

Introduction

1.1 The Smart Home

The early sci-fi envisioned homes of the future were conjured on ideas of convenience, intelligence and increased productivity. The 1950s had Robbie the Robot, the 1960s The Jetsons. These were optimistic times where the family relaxed while humanoid robots did the cleaning and cooking. Today, however, we have more realistic visions in the form of the smart home.

The term *smart home* was coined by the National Association of Home Builders (NAHB) in the early 1980s after it set up a group to push for the adoption of smart technologies in the design of new homes [1]. Since then, research into the area has mainly focused on the development of example smart houses, which showcase the possibilities that this type of technology could bring. This research along with the efforts of the NAHB and others, has yet to deliver the smart home life envisioned by many during the late 20th and early 21st century. Most of the western world is still living in homes more similar to their grandparents than those conceived by the early smart home pioneers [1].

A smart home is a domestic residence that incorporates devices to control features of the home. The most basic feature of a smart home is the ability to control devices

remotely or automatically [2]. Originally this may have been limited to switching the heating or lighting on or off, but as technology has improved almost any device can now be controlled remotely or automatically. Smart homes are now capable of incorporating large amounts of computing power to monitor the activities of its occupants and anticipate their wants and needs. They can provide users with complex customisation options allowing them to tailor their environment to precise requirements [3].

Until quite recently, it has been this vision of smart home automation that has been the focus of academic research. Over the past decade a number of large smart home projects have demonstrated the possibilities that the technology can bring to our homes [4-8]. However, the notion that a home can be a fully autonomous space, responding to users with little or no input, has not transpired [9]. Within the research literature, it is now difficult to find a smart home research project that is currently active. Furthermore, the ambitious smart home future outlined by these projects has not emerged.

Even so, the home of today has changed from that of a decade ago, but not in the ways predicted by previous smart home research. Technologies such as Wi-Fi and 3G have become ubiquitous, enabling connectivity and data transfer to occur anywhere within the home or wider world. This has led to the concept of the *Internet of Things* (IoT), whereby physical *objects* and *things* have a presence on the internet, enabling them to gather and share information with the purpose of reaching a common goal. It is this connectivity that has also brought about the most visible change in the home; in the way we consume media and entertainment [9]. Sitting down and consuming television via the internet or reading a newspaper online is now familiar to many. This has led some to move away from the term smart home, preferring to use *the connected home* as a more accurate description for the future of domestic life [9].

The technology we use in our home has also become more personal and nomadic. Smartphones have provided an *always-on* personal computer that is rarely out of arms reach. They can be used to download *apps* offering an endless array of extensibility. Software can be developed that can interface with a many different devices, protocols and technologies. They also provide a mobile window into the world of social networks

such as Facebook and Twitter. News, events, emails, iMessages, bank alerts, weather alerts and location information can be instantly received at any time of the day.

It is through the smartphone that the future smart home may be realised. With such a personal, intuitive and well-connected device, it is logical to think that it will play a critical role in the future of smart home research. However, before postulating about this future, it is important to first understand why the smart home is a useful and interesting field of research.

1.2 Benefits

The benefits for smart homes are numerous, but can be divided into four distinct categories, these being [1, 10, 11]:

- User convenience, including comfort, instant control, etc.
- Assisted living
- Energy saving
- Security and safety

Although this is not an exhaustive list of areas, interest in smart homes has generally been associated with one of these categories. User convenience has long been associated with smart technology while energy saving and assisted living have more recently seen an increase in activity [12-15]. Security and safety is often linked with the elderly or disabled as a means of providing private home support at a reduced cost [5].

1.2.1 Convenience

Convenience, in the context of smart homes, is the ability of the home to sense and react to various relevant stimuli. This reaction may be to present the user with information or to automatically control an element of the home. An example may be the automatic illumination of the entrance hall when the front door is unlocked, or for the washing machine to inform the user when its cycle is complete. The Gator Tech Smart House [4] showcases many convenient smart home technologies. Although the home is specially designed for the elderly and disabled, with an overall goal of creating assistive

environments, it demonstrates the type of convenience available within a smart home. The house boasts an impressive array of smart devices that are connected to the home's network and are designed to monitor aspects of daily life. Examples of these devices include a smart mailbox, capable of sensing when the post arrives, and a smart front door complete with RFID tag for keyless entry. The smart bathroom is capable of detecting low lavatory paper, lavatory flushes, occupants cleanliness as well as regulating the shower water temperature. Along with a whole host of other devices, the home is capable of regulating itself and responding to residents, so it can, for example, replenish stocks when necessary or automatically control the home entertainment system depending on your location. The predicted benefit of all this technology would be similar to the adoption of home appliances during the 1940-1950s. It would free people from the last remaining mundane aspects of running a modern home [1].

Superficially this type of convenience may be appealing to many. Having the lights respond to voice commands, or the heating automatically regulated throughout the day, is the type of convenient technology many still consider to be curious, if not remarkable. However, this type of domestic environment has been possible for many years [7]. What is interesting from the literature is not that this type of technology-enabled convenience exists, but rather, that users seem uninterested in adopting the technology [10]. Even amongst the small group of users who have adopted, the frustration that surrounds many automated tasks is prominent [16]. Convenience may be a good idea, but when the user struggles to understand how the technology works or is controlled, the fundamental concept of promoting user convenience can be easily lost.

1.2.2 Assisted Living

Kidd et al. describe The Aware Home, developed by Georgia Institute of Technology, which *"focuses on the computing needs of people within their everyday lives, specifically, the part of their life not centred around work or the office"* [5]. Kidd et al. identify assisted living as a specific application for smart homes. With a baby boom generation fast approaching retirement age, smart homes could provide them with cheap affordable home support and health care advice. Governments are beginning to see smart home technology as a viable option in reducing the financial burden of supporting

this generation through their retirement. The current round of EU ICT research funding includes a specific call for projects relating to *ICT for Ageing and Independent Living* and *ICT for smart and personalised inclusion* [12]. Raad et al. have developed a sensor-based system that utilises telecommunication technologies to provide a cost-effective telehealth system for the elderly and disabled [17]. The aim of this system is to *"provide a continuous communication link between patients and caregivers and allow physicians to offer help when needed"* [17]. This type of system could allow the elderly to stay within their own home while providing them with essential care at an affordable cost.

Any improvement in the wellbeing and comfort of the elderly is something that should be promoted. It may transpire that dedicated assisted living environments and systems, such as those being designed by Kidd et al., may provide the accessibility required. However, the recent functionality of the smartphone has provided an alternative platform for health assistance on a personal nomadic level, not just for the elderly, but universally. A smartphone with basic abilities can be used to call a doctor, care assistant or friend (either via a voice call or video call). They can be used to connect to the internet and relevant knowledge systems that are available. More advanced smartphones can import data from external sensors, enabling constant monitoring of heart rate, blood glucose level, temperature, blood oxygen levels, etc. Rather than being disconnected from the user, this constant monitoring enables doctors to get real-time feedback, so they can predict and anticipate problems rather than wait for the user to report issues.

As an alternative to having a dedicated teleheath system in the home, it may be that health monitoring becomes another app running on a smartphone. In much the same way we use a personal computer, a smartphone may provide the ability to make phone calls, read the news and listen to music, while at the same time monitoring our blood pressure and heart rate.

1.2.3 Energy Saving

The need to significantly reduce carbon emissions is arguably one of today's greatest challenges. Energy efficient smart technology can be utilised within the home to offer

the user a host of energy saving ideas. The 2014 rollout [18] of smart meters will allow a host of energy saving techniques to be implemented within the home. For example, those who generate their own energy can sell it back to the energy grid. Smart meters allow feedback of energy consumption to easily be delivered to the user and supplier. Wood et al. [15] explores the idea of using energy feedback displays to inform users about the amount of energy being used by the various devices currently active within their home. With these displays motivation techniques can be used to encourage a reduction in energy usage. Darby [19] investigates the use of a TV or PC to deliver information back to the user showing them their historic consumption, daily costs and comparisons with other homes. Other systems such as smart heating systems and smart lighting systems, the idea of a single off button for the home that will turn off all but essential items, based on the inhabitant's requirements, are being explored and researched [7, 20, 21].

Within the context of smart homes, energy saving will be driven by the rollout of smart meters. However, the abilities of an individual smart meter are limited and on their own smart meters are no smarter than regular meters. The *smart* abilities come from the connectivity and data drawn between different devices and systems. The smart meter rollout can therefore be seen as a convergence between the concepts of the IoT and smart homes. It is possible, indeed likely, that this convergence becomes more pronounced over time with the concepts of IoT and smart homes frequency overlapping.

1.2.4 Security and Safety

Security and safety is a universal concern and a basic priority for many people. A study by Min-Soo et al. [22] found that nearly 43% of the elderly respondents were worried about the event of an accident involving gas and approximately 22% were anxious of intruders [22]. Smart home technology offers many benefits for preventing accidents and reducing security fears. Internet Protocol (IP) based security cameras are commonly used as a home security feature. These types of cameras can be controlled from remote locations by sending control commands allowing users to view their home from the office or on their mobile phone. Security aware smart systems are a more advanced security feature. This type of system can monitor occupants in the home, learn daily

routines and use a variety of sensing technology to analyse and flag potential security risks to the user.

1.3 Consumer Opinion

Many academic and industrial research institutions have conducted research into smart homes and their potential [1, 3]. From this literature, an understanding regarding the abilities and benefits that a smart home provides to the end user can be drawn. However, to have a broader understanding as to where future smart home research is best placed, user needs and attitudes regarding smart technology need to be reviewed.

Pragnell et al. have tried to assess the smart home market potential with a report published by the John Rowntree Foundation [23]. Pragnell et al. describe the S-curve of new technology adoption, which is *"characterised by slow take-up in the early years followed by a more rapid increase in adoption, which moves the product into the mass market"* [23].

Figure 1-1 shows an example of this type of curve and that as time goes on more households adopt the new technology.

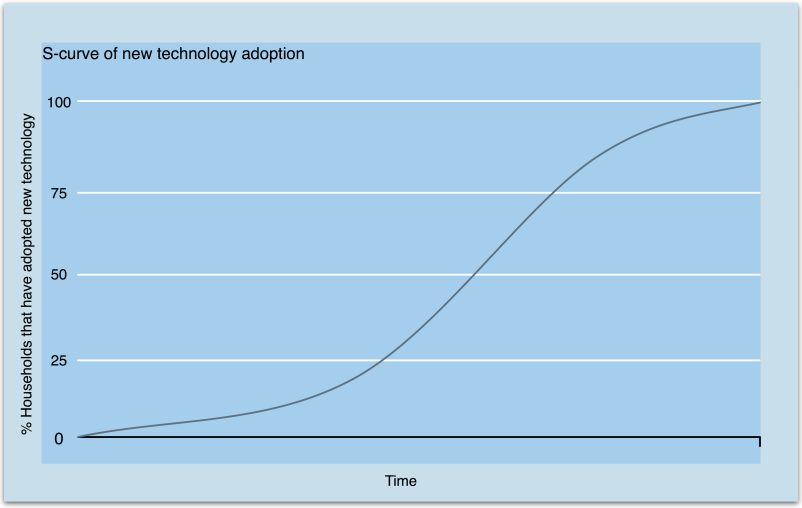

Figure 1-1 - S-curve of New Technology Adoption [23]

It is worth noting that different technologies follow different S-curve shapes, some rapidly making it to mass market, some more slowly. The time a technology takes to become adopted is therefore variable and can be affected by a number of key factors including economic, social, consumer, technological and global/political [23].

Pragnell et al. go on to describe the demand for smart homes and groups people into categories based on their interest. This interest usually relates to a person's age, with younger people more interested than older people. Although accurate at the time, these categorisations are no longer reflective of societies attitudes towards technology. Mitzner et al. demonstrate this change in attitude, especially among the older subset of society [24].

Mitzner et al. found that, contrary to the stereotypes of being afraid or unwilling to use technology, older people's "*positive attitudes (i.e., likes) outnumbered negative attitudes (i.e., dislikes), suggesting that older adults perceive the benefits of technology use to outweigh the costs of such use"* [24]. More interesting Mitzner et al. also found that technology use was most prevalent in the home.

Graham found that, rather than being age related, education is the most salient divider amongst attitudes towards technology [25]. Someone who has taken some form of higher education, for example, would have a more positive attitude towards technology and its use as a social improver. Other demographic factors affecting attitudes towards technology were age and ethnicity [25]. Overall, however, the literature shows that positive attitudes and interest in technology is increasing.

If the demographic range of society who are interested in smart home technology has increased, has this interest translated into a larger uptake of the technology?

An ON World report [26] predicts that by 2012 the smart home market will be worth $2.8 billion. According to an iSuppli[1] estimate the worldwide consumer electronics Original Equipment Manufacturer (OEM) revenue in 2010 will rise to $317.3 billion [27]. This means that the smart home market represents less than 0.1% of overall

[1]iSuppli are a commercial market intelligence organisation

spending on consumer electronics. The 21st century question that faces the smart home industry is not what is possible with the smart technology, but what is preventing the technology from being adopted by the average consumer. With over three decades of development, people are still unwilling to invest.

1.3.1 Lack of uptake

The problem of adoption cannot be pinned on one singular smart home feature, but with a host a problems. These can be categorised into 4 key areas [10, 16, 24, 28, 29]:

- Cost
- Interoperability
- Retrofitting
- Usability

The issue of cost has long been a problem for smart home manufacturers. The problem being that replacing all non-smart devices in a home, and installing the associated cabling is extremely expensive. Technology progression has begun to impact the problems associated with cost. Wireless communications standards such as Wi-Fi [30] and ZigBee [31] could, for example, reduce the need for cabling in specific situations.

However, as technology addresses one problem it creates another – interoperability. The many competing wireless and wired control protocols that exist (e.g. X10 [32], Lonworks [33], Zigbee [31] and KNX [34]) create issues with connectivity. The movement towards universal connectivity rather than individually designed and controlled smart devices can only work when manufacturers use open and common standards. Currently manufacturers are using proprietary methods and if users wish to connect different technologies they need to use middleware to bridge the different technological gaps. If universal connectivity did happen, it would allow users to bundle together different products, bought at different times, allowing them to invest slowly in various smart technologies, safe in the knowledge that the next product they buy will still connect with existing purchases. This type of approach may be possible if research

into the IoT provides open and common standards for connecting devices to the internet.

Retrofitting is the task of installing smart home technology into an existing home. At present the technology for a smart home is usually installed during construction of the building or during a major renovation. Expensive cabling, planning and simulation is undertaken to ensure that bespoke designed systems and devices are connected together [2, 35]. An individual rarely undertakes this type of building work and therefore the technology is almost never found in older buildings. Construction companies are also unlikely to budget for unnecessary building costs in all but luxury domestic buildings, resulting in many new builds still being constructed without the ability to even accommodate new smart technology. The potential for smart home retrofitting has improved with the development of wireless network devices and power line smart transceivers platforms such as ZigBee [31] (a wireless standard designed for low power consumption, latency and data rates) and LonWorks [33] (a wired network standard for controlling applications). These types of technologies are useful for sending small amounts of control data between devices but are unable to cope with the high data rates of multimedia applications, such as high-definition movie streaming. Feasibly Wi-Fi could be used for these high bandwidth applications; however, the high power consumption required for Wi-Fi transceivers generates a new set of issues.

The usability of a smart home defines the way an end-user will interact with various smart devices and control interfaces. It is an area that has, until recently, been largely overlooked. One of the major issues with smart home technology is its complexity and difficulty of use. Users often find the interfaces and control paradigms confusing to understand and frustrating to operate [16, 24, 28].

Jeong et al. researched the smart home interface preferences between U.S and Korean users. They found that *"respondents preferred to interact with a smart home using a physical device (a computer, cell phone, or remote control) rather than through communication modalities such as speech or gesture"* [36]. From the results of the survey, Jeong at el. recommend that interfaces should be adapted to a particular culture and go on to propose culture-specific guidelines.

Nylander et al. go further, demonstrating that, given the choice, users often chose to use their mobile phones rather than a computer for accessing the internet or watching videos in their home. Reasons for this choice were speed, convenience and because the phone is always on and within close proximity.

Given that users prefer a physical device, such as a phone, compared to other communication modalities such as speech or gestures, smart home usability research should focus on improving the complex user interfaces and control paradigms associated with this type of device. Poor usability has long posed a problem to the uptake of smart home technology. If a user is unable or unwilling to use or operate the technology being offered for integration in their own home then they are unlikely to invest.

1.4 Research Contributions

As will be discussed in the subsequent chapters, of the four issues raised (cost, interoperability, retrofitting and usability), the key area of usability will be the focus of this thesis. Within this area, the lack of prototyping and evaluation frameworks presents a significant bar to the future development of intuitive smart home control interfaces. A framework is designed and implemented as a novel approach for the design, prototyping and evaluation of mobile interfaces for domestic environments. This framework is intended to empower researchers and designers with the tools to facilitate the improvement of smart home control interfaces. Within this framework the specific research contributions of the author are:

1. The Reconfigurable Multimedia Environment (RME) - a physical evaluation and observation space for conducting user centred research [37, 38].
2. Simulated Interactive Devices (SID) - a video-based development and control tool for simulating devices commonly found within a smart home [39]
3. iProto - a framework component that facilitates the production, deployment, real-time analysis and round-tripping of high fidelity prototypes for mobile touch screen devices.

4. Extensible Feedback Mechanism - a tool for controlled text based two-way communication via configurable GUI plugins.
5. Visual Matrix - a visual-based video routing paradigm and software layer [37, 38].

This framework is evaluated as a round-tripping toolchain for prototyping smart home control and was found to be an efficient process for facilitating the design and evaluation of such interfaces.

1.5 Thesis Structure

Chapter 2 is divided into the seven main sections. The first two sections introduce smart homes and present background material on the area. The third section discusses in detail the reasons for lack of adoption, identifying usability as a key reason. In the forth section smart home control is discussed, in particular the abilities of the smartphone. Current approaches are discussed along with their limitations and the need for a new approach is identified in the fifth section. In the sixth section simulation is discussed as a means of evaluating smart home control interfaces without the need to build a physical home. Current approaches are also considered; however, these prove to be inadequate for the type of evaluation envisaged. The final section introduces the research problem for this thesis.

Chapter 3 is divided into two main areas. The first introduces the need for the framework as a novel approach for the design, prototyping and evaluation of mobile interfaces for domestic environments. It also gives an overview of the main framework components. With a reflection on the current literature, the second section discusses the aims of the framework allowing for the creation of a set of requirements.

Chapter 4 details the design and implementation of the Reconfigurable Multimedia Environment (RME) [37, 38] – an evaluation and observation space for conducting user centred research. It is divided into three main sections. The first outlines the role of video within an observation and capture system. The second details the physical space in which the RME is implemented. Finally the development of a new visual routing paradigm and control software is discussed.

Chapter 5 is divided into two main sections. The first details the design and implementation of Simulated Interactive Devices (SID) [39] – a video-based development and control tool for simulating devices commonly found within a smart home. The second section details the Extensible Feedback Mechanism (EFM), a controlled two-way visual communication tool.

Chapter 6 details the design and implementation of the final framework component, iProto. It discusses an approach taken for supporting designers and current workflow methodologies while allowing for the production of high fidelity interactive prototypes for smartphones.

Chapter 7 focuses on a case study to validate the framework through the creation and round-tripping of a prototype for smart home control. It includes the design, evaluation and iterative prototyping of a smart home control app with the use of the SID and iProto framework components. In addition a user study is presented detailing the standalone use of the RME with 3rd party research.

Chapter 8 concludes the work carried out in the thesis by detailing the achievements and possible limitations of the work. Ideas and suggestions for future work are also described. The future of smart home control is discussed, as are the abilities of the framework to be applied to future forms of smart home control.

In summary, the main objective of this thesis is the development of a framework that can be used to facilitate the design, prototyping and evaluation of smart home control interfaces. The novel approach taken considers the current literature and established theories in the smart home, prototyping and simulation domain. The framework is tested to assess whether it conforms to the requirements and to justify if it can be used for its intended purpose.

Chapter 2

Literature Review

This chapter outlines the main areas relating to this thesis: smart homes, prototyping and simulation. The first two sections introduce ubiquitous computing, the Internet of Things (IoT) and smart homes. A number of previous smart home related studies are also discussed. The third section discusses in detail the reasons for lack of adoption, identifying usability as a key reason. The forth section focuses on smart home control outlining the smartphone as being the key smart home control device. Current approaches to smartphone prototyping and their limitations are discussed in the fifth section. In the sixth section simulation and the Wizard of Oz technique are introduced as a means of evaluating smart home control interfaces without the need to build a physical, functioning home. Here, the Wizard of Oz technique is outlined as a method for fooling a user into thinking they are interacting with a computer system when they are, in fact, interacting with a human operator or *wizard.* Finally, the research problem and focus of this thesis is outlined in the seventh section.

2.1 Introduction

A smart home is the physical realisation of a ubiquitous computing system. It is a place where technology and everyday life converge. In 1991 Mark Weiser was the first to describe a vision of a new computer revolution of a world, not of desktop computing,

but of ubiquitous computing. Weiser wrote, *"the most profound technologies are those that disappear"* [40].

Ubiquitous computing is a vision of nonintrusive computing whereby the availability of computers throughout the physical environment has become such that they are virtually, if not effectively, invisible. We are currently living in a time where ubiquitous is becoming a reality. A decade ago we generally viewed computers as desktop machines that store applications and the data they produce. This is the antithesis of ubiquitous computing. In a world of ubiquitous computing, the computer would become embedded into walls, clothes, light switches and cars [41]. It would become so natural to use, so fitting into our environment, that it would be indistinguishable from it. It is a world where computers are not attention-demanding boxes but sophisticated technology-enhanced environments requiring so little attention that a user can interact with the technology without thinking about it.

The belief that the ubiquity of computers would render them invisible to the user was profound. When Weiser penned his ideas on ubiquitous computing, the internet was in its infancy and computers were still relatively immobile. To a degree, Weiser vision is now becoming a reality. Computers *are* becoming increasing small and ubiquitous, so as to invisible to the user. Ovens, fridges, speakers, even traditional copper cables [42] now come equipped with some form of computing device.

Weiser and his team of researchers began investigating ubiquitous computing as a radical answer to what was wrong with the personal computer, described by Weiser as *"too complex and hard to use; too demanding of attention and too dominating as it colonised our desktop lives"* [43]. Later he would go on to describe *calm computing* as the goal, describing the desired state of mind of the user [43]. The goal of calm computing would be realised in an era of ubiquitous computing, the third of three stages of computing outlined by Weiser.

Although many of Weiser predictions may have been accurate, Weiser fails to predict the current reality regarding ubiquitous computing. Rather than being a catalyst of calm computing, the ubiquity of computers has led them to dominate a large majority of people's lives. The availability and accessibility of information has led society to

become a slave to the computer. Computers have touched every part of our existence, from travelling to bathing, and in doing so they have brought about a revolution not seen since the industrial times. However, as with any revolution, change brings about confusion. One of the major concerns regarding computers is their difficult and confusing nature [10, 16, 24]. With the number of connected devices expected to reach 50 billion by 2020 (equivalent to nearly 7 devices per person) [44], the inconsistencies, abilities and differences in interaction techniques between devices will further lead many in society to become perplexed by computers.

It could be argued that we are currently only just within Weiser's third phase of computing and that calm computing will prevail if given time. Weiser's phases of computing are outlined below.

Phase I: The mainframe era can be characterised as the era of many people sharing an individual computer. The computer was a highly specialised piece of equipment that was run by experts behind closed doors. Many people would connect to the single computer using dumb terminals. Today, mainframe computers usually exist in the form of super computers [45]. These extremely powerful computers are mainly used for computing various simulations, from weather predictions to effects of age on nuclear warheads.

Phase II: The personal computer defines the second major phase of computing. In 1984 the number of people using a personal computer surpassed the number using shared computers [41]. The characteristic of this phase of computing is the *personal* nature of the computer. A single computer with a single user means the personal element is very real. People become attached to their computers, often naming them or getting frustrated at them in the same way they would a human, and although one person may have many computers the relationship is always one computer, one person.

Phase III: According to Weiser the third phase of computing is ubiquitous computing. This phase can be characterised as the era of many computers with many users. This could be the machines we connect to for email or for cloud computing, but it may also be computers embedded into everyday objects such as cars, walls or consumer goods. The era of computing will be defined by the connectivity of devices; the *Internet of*

Things [46] will allow previously dumb objects to become smart. The sharing of information between these previously unconnected objects will become commonplace, to such an extent that we will forget their beneficial effect on everyday life. In much the same way electricity transformed the 20th century, the ubiquitous computer may do so for the 21st.

Weiser predicts that this transformation will lead calm computing [41]. However, the technology currently in use requires constant attention from a user while it bombards their senses. The aim of *calm technology* is to create devices that provide a calming experience by moving the technology to our peripheries. Weiser uses the example of driving. "*Ordinarily when driving our attention is centred on the road, the radio, our passenger, but not the noise of the engine. But an unusual noise is noticed immediately, showing that we were attuned to the noise in the periphery, and could come quickly to attend to it"* [41]

As we move further into the era of ubiquitous computing a fundamental shift in how we perceive and interact with computers will be realised. The idea of a computer as being a machine on a desk will become as foreign as Watson's famous misquote predicting *"a world market for maybe five computers"* [47]. People will come in contact with hundreds of computers on daily basis. Objects, which had previously been inert, will be connected together and made smart. This will happen at a national, local and personal level. In one place in particular, the home, the transformation could prove most profound. It is here that there is scope to make the ultimate ubiquitous computing environment, the smart home.

2.1.1 The Internet of Things

The idea behind the Internet of Things (IoT) concept is that once pervasive *things* or *objects* become uniquely identifiable they are then able to interact with each other and cooperate with their neighbours to reach common goals [48]. With this idea, each *thing* or *object* would be capable of generating or computing data, then exchanging the information over the internet.

The Internet of Things paradigm includes many visions and, although the previous definition holds true, there is an increasingly broad array of definitions that can apply to the concept. This is partly the result of the name *Internet of Things* that syntactically is composed of two terms. *"The first one pushes towards a network oriented vision of IoT, while the second one moves the focus on generic objects to be integrated into a common framework"* [48]. A third, semantic oriented vision of IoT is derived from the ability of each object or thing to have a unique address and exchange of information. The broad scope of vision that can be applied to IoT is shown in Figure 2-1.

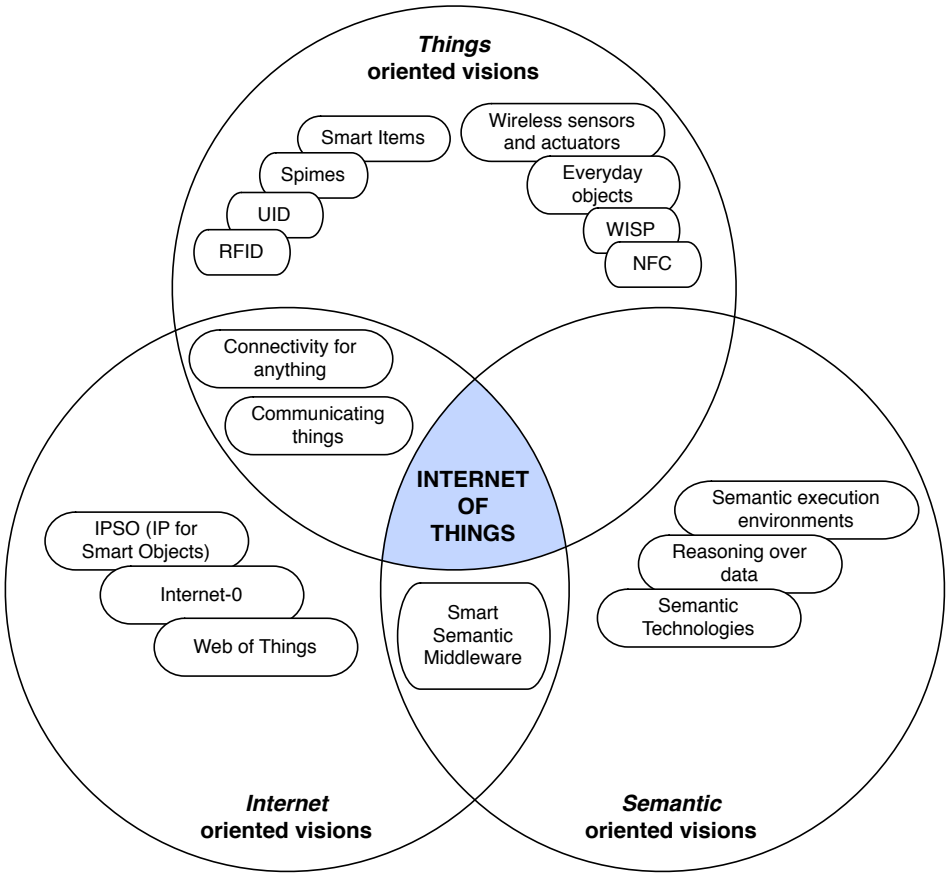

Figure 2-1 - Internet of Things paradigm as a result of the convergence of different visions [48]

Relating Weiser's vision of the ubiquitous computing to IoT, there are synergies; however, the two concepts are significantly different. Ubiquitous computing does not imply any use of objects, nor does it require any use of the internet. Rather, it is about

seamlessly integrating technology into our lives. It may be that IoT is a building block for this seamless integration; however, as a singular concept IoT does not have this integration as its end goal.

IoT will enable the physical world to have a presence on the internet. It will allow previously dumb objects to become smart, enabling a service architecture to be built upon the array of data generating hardware. Once this presence is established applications will be able to take advantage of the data and information provided, ideally in a seamless fashion.

The terms: ubiquitous computer, the Internet of Things and the smart home, all describe different concepts. However, as we move into an evermore connected society the boundaries between these individual terms will begin to blur. A smart home may be nothing more than a collection of smart objects – these objects being part of a wider Internet of Things. This may, in turn, produce a calm ubiquitous computing environment, similar to the vision outlined by Weiser [41]. This thesis will, however, focus on the smart home.

2.2 Smart Homes

Harper defines a smart home as *"a residence equipped with computing and information technology, which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond"* [1]. Through this technology management a smart home should strive to adapt to the occupant, improving their wellbeing and life experience. Ma et al. [49] define a *smart space* by three core essential features. First, it is a physical environment equipped with electronic devices and embedded systems. Second, it must have, at some level, the abilities of perception, cognition, analysis, reasoning and anticipation. Finally, it aims at adapting to humans rather than the reverse [49].

A smart home should not be a frustrating computer rich environment that removes control from the occupant, thereby creating a fully automated uncontrollable system. It should not be an environment that is born solely because the technology that permits it

exists. A smart home is still a *home* and although it may incorporate a large amount of technology, this should be presented to the user in a controllable and non-confusing manner.

The concept of the smart home only really entered popular culture during the 1990s. Before then it was mainly the province of home hobbyists and science fiction writers [1]. By the late 1990s the concept was well established. However, even today only a small number of smart homes have been built and these are mostly experimental projects [4-6]. An even smaller number of commercial smart homes have entered the market.

Research into the area has mainly centred around these experimental projects with alternative research being limited. At the turn of the millennium Hindus comments *"technology in homes has to date received little attention within the research community"* [50]. This is not to say that the research literature is insufficient or hasn't moved forward since Hindus made those comments, more that it can at times be fragmented, spanning a larger scope of subjects compared to other technology related areas [1].

To date smart homes have mainly been developed by large research institutions [4-6] or commercial research and development (R&D) projects [51, 52]. Developing systems to control various aspect of a home is a complex task. This is mainly due to the overhead of connecting and controlling various devices and sensors deployed throughout the house that enable the home to adapt to the user. Smart houses have been, in general, developed for research investigating when and how a home should respond to its user. Three of the larger research-based smart homes are introduced below to give an idea of the functionality and focus showcased in current literature.

2.2.1 The Aware Home

The Aware Home began in 1998 and was developed by Georgia Institute of Technology [5]. Researchers on the project have continually published research relating to smart homes and smart technologies from conception and are still actively exploring this area today. The aim of the home is to focus on the computing needs of people within their

everyday lives, specifically, the part of their life not centred around work or the office. The home itself has two identical and independent living spaces consisting of two bedrooms, two bathrooms, one office, kitchen, dining room, living room and laundry room along with a shared basement with home entertainment area and control room.

Having two independent living spaces allows inhabitants to live on one floor while prototyping and demonstrations happen on the other. The research conducted at the home can be split into three areas. The first is focused on application development and evaluation. Within the home an array of sensors is being used to help interpret and understand the contextual clues of its occupants. Advances in sensor technology need to be matched by the rapid development of applications that can use the information provided. These types of applications, which can be called context aware applications, can be used, for example, to automatically control heating or lighting systems or to find lost objects within the home. To assist the rapid development of these types of applications a software infrastructure has been built.

The second area addresses the question of what purpose the technology within the home serves from the occupant's perspective. An example of this is support for the elderly. As the baby boom generation ages more people require support that cannot be provided by younger generations [5]. Researching systems capable of providing monitoring services similar to those currently supported by assisted living centres, will potentially allow more elderly people to stay in their home while still receiving the level of care they require, at a minimum cost.

The final area is evaluation and social issues. This involves evaluating issues such as privacy and how the data collected will be stored and used in such a manner so as to not breech the occupant's privacy.

The areas being researched at the Aware Home are addressing many different areas of ubiquitous computing. Although many useful research results have arisen regarding the use of the home, having an increased focus on one area of ubiquitous computing, such as usability, might yield even greater results. The original purpose of the home has evolved from a space to investigate the use of context aware applications to support

people (especially the elderly) in their home lives, into a *"multidisciplinary exploration of emerging technologies and services for the home"* [53].

Figure 2-2 - Georgia Institute of Technology Aware Home [54]

The Aware Home is almost an ideal space for smart home research. However, the only real common thread in the research is the original goal of supporting the elderly. Other research themes seem disparate, wide ranging and at times not requiring the use of the smart home at all (e.g. "*Virtual Rear Projection: Do Shadows Matter?"* [55]).

Another issue with the home is its cost. Building the home from scratch gave the researchers power over every aspect of the design resulting in a highly bespoke, but very financially expensive home. The project was funded with a $700,000 grant from the Georgia Research Alliance [56]. The Aware Home has produced a lot of novel and valid research and that in many ways justifies its cost. However, replicating such a home is extremely difficult for all but large research and corporate institutions. The home focuses on automation and the development of the relevant infrastructure to enable many complex bespoke systems to communicate and respond to occupant

interaction. It is difficult to quickly set up the home to evaluate small, but increasing relevant, usability issues.

It could be argued that real power of the home lies with the ability to observe people in a smart home environment while they engage with smart devices. In the Aware Home this is achieved with a complex system designed to capture events based on the triggering of sensors. For example, if the kitchen is used, RFID tags worn by a subject and pressure sensors in the floor, trigger the capture of snap shots from a continuously running video stream [57, 58]. This type of system is therefore aimed at longitudinal studies. The system could potentially be replicated with a human at a significantly reduced cost and complexity for shorter evaluations. This has been demonstrated with the development of the Reconfigurable Multimedia Environment [37, 38] and Simulated Interactive Devices (discussed further in chapters 4 and 5 respectively). If the power of the Aware Home's observation system could be replicated at a reduced cost, whilst also being tailored for user-based research relating to smart home control interfaces, the resulting knowledge gained could have a profound effect on our understanding of users and their ability to control smart home environments.

2.2.2 Gator Tech Smart House

The Gator Tech Smart House is a programmable ubiquitous space [4]. The first generation of ubiquitous environments were generally closed off systems unable to adapt to the newly developed technologies [59]. This problem meant that once a home was considered complete it became very difficult to install new devices or systems within that home. The Aware Home, for example, was designed to support a specific set of equipment and devices. Although it is possible to modify the house it is very difficult. The Gator Tech Smart House tries to address this limitation [4]. The home is specially designed for the elderly and disabled with an overall goal of creating assistive living environments.

The house exhibits an impressive array of smart devices (as shown in Figure 2-3) that are connected to the home's network and designed to monitor all aspects of daily life. Examples of these devices include a smart mailbox, capable of sensing when the post

arrives, and a smart front door complete with RFID tag for keyless entry. The smart bathroom is capable of detecting low lavatory paper, lavatory flushes, occupants cleanliness as well as regulating the shower water temperature. Along with a whole host of other devices, the home is capable of sensing itself and the residents, so it can for example, replenish stocks when necessary or contact external help when in need. Behind the scenes this has been made possible due to the development of a generic reference architecture, which is applicable to any ubiquitous computing space [4].

The architecture breaks the system down into a number of separate layers: physical, sensor platform, service, knowledge, context management and application. To achieve the programmable ubiquitous space the sensor platform can communicate with a number of different devices, appliances, sensors and actuators and can represent these to the system in a uniform way. The sensor platform effectively converts any sensor or actuator in the physical layer to a software service within the service layer, thus decoupling them from the physical world. This technique allows new technology to be installed into the home as it becomes available.

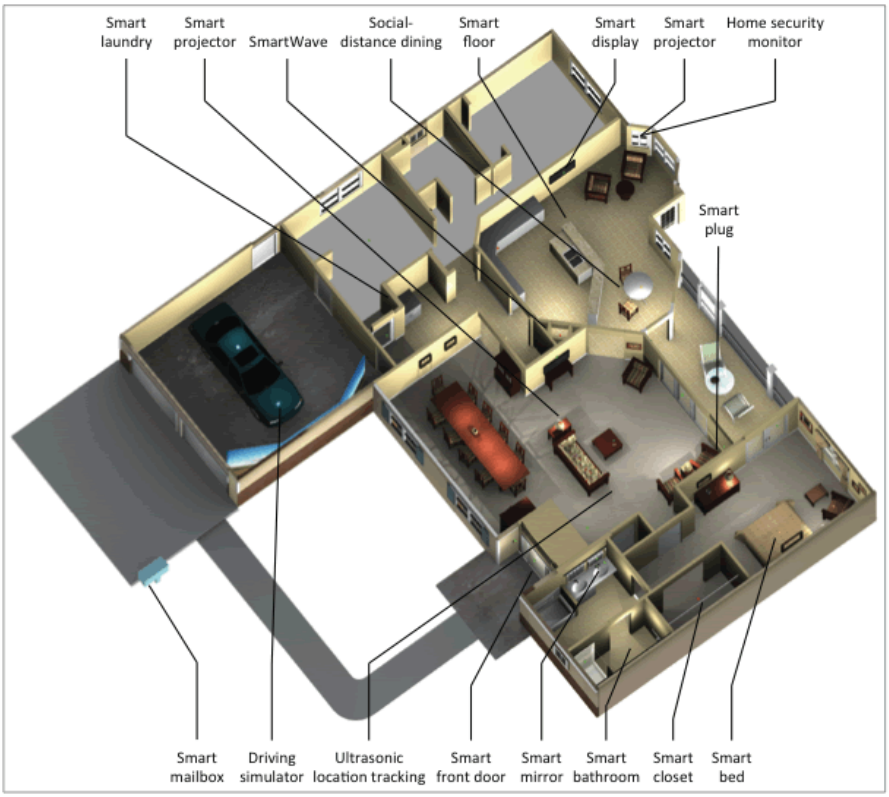

Figure 2-3 - Gator Tech Smart House [60]

Using this type of approach, the Gator Tech Smart House attempts to overcome the issues relating to future technology integration. It has specific emphasis on a middleware architecture that can be used to extend the home through various additional sensors and components. This includes mobile devices, specifically a smartphone, in addition to the smart devices discussed previously. The use of a smartphone is identified as a *"magic wand for the home"*[4]. This middleware is important as it enables the home to better adapt to the constantly changing technological landscape. It also increases the longevity and functionality of the home. Wide ranging experiments can be performed more quickly without a redesign of the home or the underlying software architecture. The Aware Home implemented a rigid design (with any inclusion of additional components requiring retrofitting and the use of a bridging layer), whereas the Gator Tech Smart House strives to accommodate for a wide range of devices. It is

especially important that any environment used for conducting smart home usability research incorporates this element of extensibility.

The home still has similar limitations to the Aware Home, in that it is an extremely complex bespoke environment that has been designed and produced as a single entity. The developers have tried, where possible, to use low-cost devices and components, but the overall complexity renders the home very costly. This reflects the aim of the space as a test bed for researching how those smart home technologies can be integrated together. However, it is due to the cost, complexity and rigidity of the design that it would be difficult to use the space for research that required rapid changes in the environment – such as those required for smart home control research.

For example, the smartphone was identified as a useful control device, but research focused on the communication mechanisms surrounding the connection of the phone to the home through a software communication layer. To address issues relating to smart home control, research would have been better focused on issues such as user interaction, rapid interface development and the accommodation of other forms of interaction. The constraints placed on the environment limits the use of the Gator Tech Smart House, and the related systems, for investigating this type of research.

2.2.3 The Ubiquitous Home

The Ubiquitous Home developed by the National Institute of Information and Communications Technology in Japan created a real life living experiment that involved a couple living in the home for 16 days [61]. The home was a test facility for the creation of new services by linking devices, sensors and appliances. The layout is similar to the Aware Home and Gator Tech Smart House; it has a living room, dining-kitchen, study, bedroom, washroom and bathroom. However, unlike the two other houses the user interface of the Ubiquitous Home is in the form of a robot called Phyno.

Phyno comes equipped with a camera, microphone and speaker and is capable of speech and face recognition. It can be used to control various aspects of the home, for example when a user wants to turn on the television they would just ask Phyno. Another example is when the user wants an idea for dinner, they can tell Phyno the desired food and it

will bring up recipe ideas. Interacting with an environment in this way can create a more natural and realistic experience than using a traditional control device such as a PC. However, frustration became a major problem when Phyno misunderstood a command or malfunctioned. Within the 16-day experiment Phyno was out of service for 2 days and its audio-visual recognition was sometimes extremely poor [61].

Figure 2-4 - Phyno [6]

Using a robot for a home interface can be novel, but it is not currently a realistic option for controlling a home. The current available technology cannot fully support this type of interface. This is demonstrated by the poor voice recognition and an inability of Phyno to stay functional during the relatively short 16-day experiment. Other robots of this type also suffer from this inability to stay functional. For example, Asimo, [62] a humanoid robot developed by Honda, frequently requires human intervention during normal operation and if it breaks down a team of engineers are required to have it operational again. The idea that Phyno could become the robotic home butler counters the ideas put forward by Weiser that ubiquitous computing systems should be calm, intuitive and in the background. It may be that in the future humanoid type robots do integrate into our lives in a calm and ubiquitous manner; however, in the near future this form of interface will induce the opposite response. Having an automated control mechanism that constantly misunderstands instructions can be worse than using a slower manual process.

Compared with the Aware Home and Gator Tech Smart House, the Ubiquitous Home has an increased focus on the usability of a smart home. The space includes numerous ceiling mounted cameras and microphones for capturing occupants interacting with Phyno and the levels of frustration or pleasure they express. However, the research emphasis is on an environment solely orientated around Phyno. This has led to the development of a space largely customised to enable a robot to have control. The environment could be used as a reference for a more generic evaluation environment; one designed for various forms of smart home control interfaces. However, in its current form, the tight integration with Phyno limits the use of the Ubiquitous Home in alternative smart home control research. As with the Aware Home and Gator Tech Smart House the Ubiquitous Home also makes no attempt to enable changes to the type of control or the interface used. Rather, it is an assessment of a unique form of interaction – of a type that is frustrating and limiting.

2.3 Why is my home not smart?

In section 1.3.1 the key areas preventing smart homes from entering the mass market are introduced. The areas being [10, 16, 24, 28, 29]:

- Cost
- Interoperability
- Retrofitting
- Usability

Although each area does hinder the uptake of smart homes, some areas have a greater significance than others.

2.3.1 Cost

The cost of any solution is always a key issue and it is especially important when that solution is aimed at the general mass market. When consumers are given the choice between a traditional home and a smart home, with the cost of the latter being significantly inflated, the financial practicality of the traditional home usually takes

precedent. Because of this, home developers are unwilling to install the expensive equipment required to take a home into the smart realm, with the understandable attitude that they might have an expensive asset on their books that is extremely difficult to sell.

However, it is not the case that consumers are not interested in expensive electronics and services [63], in fact if anything can be taken from the last 30 years, and the explosion in value of the consumer electronics market, it is that people like buying electronics. This may be down to consumers purchasing large amounts of cheap electronics, but considering one the worlds most valuable company is Apple Inc. [64]; the producer of high-end computers and mobile devices, the argument can be made that if consumers desire a product, they are willing to spend significant amounts of money to obtain it.

Apple have achieved their success by not only making their products fashionable, but by also creating a great user experience that is strikingly different from most other manufacturers. When producing products Apple takes a modular approach to each aspect of the device, ensuring that above everything else the interface does not confuse the user. The cost of Apple products is sustained because people are willing to pay for the simplicity and quality design incorporated into Apple devices.

Of course, many factors can influence the purchasing decisions of consumers, from current fashion trends through to outright necessity, and although lowering the price of smart home technology may help stimulate uptake, it is not the ultimate answer. This is especially true of smart homes where, rather than the cost of the technology, there is a fundamental lack of user understanding and overall confusion of what the technology can do and how it can easily be controlled.

It is only when this issue of confusion is addressed that the technology will become more widely adopted. Part of this confusion is down to a lack of interoperability between devices [16]. From a user perspective this clouds what can and cannot be controlled within the home.

2.3.2 Interoperability

Interoperability is the ability for a heterogeneous set of devices or systems to exchange and use information. A significant portion of technology lacks this capability and this is most prominent when rival manufacturers develop or adapt different technologies (e.g. HDDVD vs. Blu-ray), with the hope of cornering the market. Current smart home manufacturers have developed control protocols such as X10 [32, 65], Lonworks [33] and KNX [34], which allow different devices to communicate with one another, and although this is beneficial for consumers, the fact there are three competing non-interoperable control protocols illustrates the underlying problem. Non-interoperability can develop in two ways:

- Naturally – the same idea is being independently implemented in different
- Artificially – the same idea is copied and implemented with restrictions set in place to prevent cross communication

Both are usually exploited by manufacturers as a way of locking consumers into a particular technology and preventing them from sourcing additional devices from a different manufacturer.

Issues surrounding interoperability can be overcome by using 3rd party bridges, or middleware, that join various devices or protocols together. These are often complex systems that have to be configured on an individual basis, thereby creating another layer of complexity for the end-user. This results in a non-unified user experience and an increase in the required computer literacy.

Consumers should have the confidence that different products, purchased at different times, will interoperate. Without the use of middleware, the movement towards universal connectivity can only happen when manufacturers use open and common standards. This seems unlikely, as manufacturers seem intent on locking customers into their technology. However, as technology increases in functionality, the use of bridges may become more ubiquitous. The current generation of smartphones provide a platform for the development of many bridges across a wide range of communication technologies and devices. With this type of device users may soon be able to purchase

various smart products at different times, spreading the cost over a long time scale, safe in the knowledge that future purchases will still connect with existing ones.

2.3.3 Retrofitting

Although a modern home now holds numerous complex electronic devices the home itself can still be regarded as a traditional home. The physical characteristics are still very similar to a home from the 1950s. For the majority of these traditional homes the only realistic way to incorporate the technology required to make them *smart* is to retrofit it. Retrofitting is currently a complex process that usually involves a dedicated company specialising in such a procedure. This company will analyse the home and all the user requirements then install a complete end-to-end solution. Although convenient, these solutions are expensive and often have various interoperability issues [66].

The need to install complex equipment into the heart of the home has led to this situation. Expensive cabling, planning and simulation are often required to prevent the extensive amount of equipment performing incorrectly. It is also physically difficult to change a home so it can accommodate new technology. Pulling up floors, drilling through walls and setting aside space can be a challenge for the average home occupant. These issues have recently become far easier to address with the ubiquitous nature of Wi-Fi; however, installing smart devices is still hindered with the issues surrounding interoperability. This prevents users from upgrading parts of their home with equipment from different manufacturers or spreading the cost of the various smart devices required because there is no common control protocol. Feasibly this could be overcome with wireless standards such as ZigBee. However, unlike Wi-Fi, this has yet to become a ubiquitous technology.

2.3.4 Usability

Usability is an issue affecting the whole life span of a users interaction with a smart home, from unpacking and installing a device through to day-to-day and extended long-term operation.

It becomes a particular problem with smart home interaction due to the intimate nature between a user and their home. The movement towards Weiser's third phase of computing; ubiquitous computing (discussed in section 2.1), means smart technology is becoming more invisible. Therefore, traditional notions of user interaction (between a user and a desktop computer) no longer cover the disparate actions performed by users within a smart home.

Within a major Microsoft study [16] involving 14 households that were given smart technology, users found the unreliable behaviour and unpredictability of both the interfaces and technology frustrating. All participants either disliked the complex user interfaces (which they found had limited use) or had trouble learning how to use them. One participant commented he disliked *"Teaching other people how to use it, the girlfriend acceptance factor is not that high"* [16].

Within the older members of society, Mitzner et al. found that the most negative attitudes were "*frequently associated with technology creating inconveniences, unhelpful features, as well as security and reliability concerns"* [24].

Within smart home research literature, one major focus has been how technology can be used to automate tasks. This automation had led to the extensive use of context aware applications that try to anticipate or predict what an occupant wants from their home, e.g. warm up the coffee machine during the morning shower. However, as demonstrated with Yamazaki's Ubiquitous Home [61] and Brush's [16] empirical study, automation can lead users to feel frustrated by smart technology. Hamill [67] shows that what users really want from a smart home is not automation, but control. Hamill discusses two rules that designers of smart domestic appliances should follow: "*the control rule, putting people firmly in control; and the non-presence rule, keeping the devices as unobtrusive as possible"* [67].

Smart home usability can be considered a key reason for lack of smart home adoption and one that is paramount when viewed against other issues [10, 16, 28, 29]. Although other problems can be addressed, they all require a major improvement in usability and a greater understanding of how control can be delivered to the user in a calm and predictable way.

Therefore determining how a user wants to control their smart home and how this is best delivered in an intuitive, easy to use manner needs to be understood.

2.4 Smart Home Control

The traditional home is controlled via numerous dials, switches and electronic embedded systems. Devices are disparate, disconnected objects, unable to communicate with each other or the outside world. Examples of such systems include heating, lighting and metering. In the traditional home, all are separate and uncontrollable apart from particular switches, which are specifically designed. A smart home strives to be the opposite of this, while in some cases adding an element of *awareness,* and it therefore requires a radical re-think into how a user will control such an environment.

With the potential to have smart objects in the hundreds, the control of a smart home could quickly become confusing or overwhelming for the user. This has already been demonstrated with the interfaces of the majority of consumer appliances [68]. As available computer power has risen so has the number of options and features available to the user, which in turn has led to complex and confusing interfaces. A user will rarely know all product features available to them; rather, they will choose to remember a core set of commonly used ones, ignoring the more obscure or infrequently used [69]. This effect has plagued the current generation of smart homes with the sheer number of options and features available.

The current generation of smart homes enable users to control the environment in a number of different ways. This can vary depending of the level of automation, assistance and control the developers have built into the home. However, as system capabilities expand, users can increasingly feel like they have lost control [70]. The technology within the home can become challenging and independent. This lack of control has led researchers to investigate different ways users can interact with their home without loosing the vital element of control. Intille [8] looks at designing a home of the future, not by reducing the technology within the home but by presenting the user with a different interface.

In the Intille home of the future the window frames include a tiny light-emitting diode that glows when the temperature rises. This indicates to the user that opening the window will reduce the temperature, while being more economical and just as effective as increasing the air conditioning [8].

Within a home environment a user will perform many tasks. These tasks will vary day to day but over time they will generally form patterns or chains. Brushing your teeth for example will usually take place once in the morning and again in the evening. Crabtree et al. describe how these tasks, or actions, occur in various *action centres* [71]. A kitchen table may become an action centre for a number of everyday activities. A user may also multitask when performing these actions; while brushing their teeth in the morning a user may also find their socks, pick out a tie or maybe check the weather for the day.

When determining what type of control interface should be used within a smart home, the choice should reflect how a user currently performs actions and tasks. Smart home control systems should enable mobility, interruptions and multitasking [72]. "*In a familiar environment, human behaviour assumes certain regularities. Doing everyday chores often turns into chains of action, which assume patterns, such as doing the laundry at a particular time and in a particular place and way* [72]. Smart home control systems should be able to adapt to the routines and action chains that develop over time. In order to achieve this aim, a control system will generally be based on one or more of the following user interfaces [72]:

- Static PC terminals or central computer
- Embedded system
- Natural language
- Mobile device

2.4.1 Central Computer

A traditional PC is a versatile form of computer. Software can easily be developed and installed and the large screen real estate allows for a variety of user interface (UI) designs. Traditionally a mouse and keyboard have been used as the main form of user

input (a mouse being replaced with a track pad on a laptop). This changed with the introduction of large touch screen panels; however, these still rely on the principles of the mouse.

A PC can provide extensive customisation compared to traditional dials and switches. The major drawback of a PC however is its size and weight, meaning it must usually be static within an environment, thus it does not support mobility. It is also difficult to support multitasking given that a user must return to a specific point to perform control tasks. This is cumbersome and restrictive resulting in a PC being a control device that a user has to adapt to rather than adapts to the user.

2.4.2 Embedded systems

An embedded system is a computer designed for a specific function. Unlike general-purpose computers such as the PC, embedded systems are usually integrated or *embedded* within a larger system and are used to deliver some form of electronic functionality. Embedded systems have become relatively ubiquitous within western society, for example, microcontrollers or digital signal processors have been embedded into everything from washing machines to mp3 players.

Embedded systems play a vital role in the current generation of smart homes as they make up a substantial amount of the computing technology used to make the home *smart*. Many of the sensors, for example, are developed using embedded systems that are then integrated into a larger network controlled by a general purpose PC. It is even possible to develop an entire smart home using nothing but embedded systems [73].

Although these systems are being deployed within smart homes, their role as a form of interaction is limited. Much of the smart home usability originated literature is investigating how the interface can be unified and adapted to the user [36, 74], rather than becoming disparate or heterogeneous, as is the case with many rigid embedded system interfaces.

Although embedded systems are essential to the development of smart homes (this is especially true of sensing and data collection), and for the IoT, their future role as a

means of control, through some form of interface, is restricted. The singular nature of embedded system interfaces often means they only fulfil a single purpose.

2.4.3 Natural Language Processing and Gesture Recognition

The use of natural language processing and gesture control interfaces allows for greater support of the common forms of human-to-human interaction. Voice commands and hand gestures are examples of everyday human-to-human communication that can be interpreted by these fields of research.

The current generation of games controllers have embraced the idea of human gesture control. Beginning with the introduction of the Wii [75] and its handheld pointing device capable of detecting movement in three dimensions (3D), the field is currently led by the Xbox Kinect [76], a camera-based 3D motion sensor, capable of tracking the movement of multiple users. The Xbox Kinect means a user can interact with the software using hand gestures rather than the traditional controller, it can even use facial recognition to remember custom preferences for individual users.

More recently the availability of voice control has taken prominence. The field of natural language processing (NLP) enables a computer to extract meaning from a natural human language. For example, Siri, an application released for the Apple iPhone [77] provides users with the ability to use voice to question and control the phone e.g. "*will I need an umbrella this afternoon?"*. Presented with this type of question, the phone will use various sources to triangulate its position, check the weather report for the afternoon and report back if it will be raining. Although impressive as a demonstration, this type of control is not particularly reliable. Siri and other voice control systems are discussed further in section 8.3.

These types of communications are not fully supported by today's computers. The Xbox Kinect only released their SDK in mid 2011 meaning developers are still trying to realise its full potential. Also the complex nature of this type of interaction means that development of such technologies is very challenging leading to large numbers of errors and mis-communication between the user and computer. Due to these major limiting

factors, this type of control is currently considered unrealistic. The technology is in it infancy and has not yet matured to overcome these limitations.

2.4.4 Mobile Device

One of the most common forms of mobile control device found in domestic environments is the infrared (IR) remote control. This familiar device has long been used to control televisions, radios, DVD players, etc. and is a key control device for many electrical items, particularly audio-visual devices. An IR remote is usually limited to issuing a set number of commands to a single specific device. It is controlled using a set number of buttons, the layout or function of which is difficult, or usually impossible to change. Due to the these limitations an average household would usually expect to have many IR remote controls each controlling a specific device within their home.

The development of the universal remote control (URC) attempted to overcome this issue by allowing the user to control multiple devices from the same remote. A URC gives a user the ability to program a function or command for each of the remote buttons enabling it to control multiple devices using a range of IR command codes. Although this is an improvement, the permanence of the layout limits the functionality and usability of IR remote controls and URCs, described by Nielsen as complex and frustrating [78]. This prevents them being used in a technology-enriched space such as a smart home where research has moved through the domain of the mobile feature phone [79] and PDA [69, 80, 81] and is currently centred on the smartphone [82].

By the end of 2012 there is expected to be more mobile-connected devices than people on the planet [83]. A mobile phone is a very personal object and like a PDA they usually support a number of communication standards, thus it satisfies the criteria that a smart home control interface needs to support mobility.

The differences between a mobile feature phone, PDA and smartphone can be defined by their screen type, connectivity, operating system and control paradigms. As shown in Figure 2-5 feature phones have monochromatic screens, are operated on the GSM network only (or similar standard) and are controlled using a set of buttons and menus. PDAs can connect many different network protocols, have colour screens are normally

manipulated with the use of a stylus. Both these types of devices are produced with software that is difficult to upgrade or provide additional software features.

Figure 2-5 - [a] Mobile Feature Phone [b] PDA [c] Smartphone

Unlike PDAs however, feature mobiles phone have also been very difficult to develop for. The operating system was usually locked down with special permissions and licences needed from manufacturers. However, all this has changed with the current generation of smartphones. A smartphone is more than just a phone, they are now sophisticated computers equipped with applications (*apps),* shown in Figure 2-6 [a], which allow a near endless list of features. It is with these *apps* that a user can run 3rd party software allowing the phone to be a control device as well as a telephone (shown in Figure 2-6 [b]). A smartphone also uses a different control paradigm for manipulation. The operating system has been specially designed to enable touch control and gesture support.

Figure 2-6 - [a] Apple iPhone Example Smart Home Control App [84] [b] Apple iPhone Showing Different Apps

Throughout this thesis any reference to an *app* is describing an application designed to run on a smartphone. Any reference to an *application* is describing a traditional application designed to run on a desktop computer.

2.4.5 Smartphone

Koskela et al. [79] undertook a research study conducted in the area of smart home control interfaces. In the study they present three different smart home user interfaces: a media terminal, a PC and a mobile phone. These were empirically evaluated to assess user preferences. A summary of this study is presented as it gives an interesting overview of the types of control devices available, which end-users prefer.

Koskela et al.'s experiment involved a non-technically orientated couple using a smart apartment (called the eHome) for six months, with their time in the apartment being

empirically evaluated. The apartment could be controlled from the three previously stated control devices.

The PC was mainly used in the living room, as was the media terminal (connected to the living room television and manipulated via an IR remote) and the mobile phone could be used anywhere inside or outside the home (dependent on a cellular network signal). The control devices were connected via a number of different networking technologies (Ethernet LAN, WLAN, GPRS) to a central computer that managed the smart objects.

The PC control device used a graphical user interface (GUI), as shown in Figure 2-7 [a], with a direct manipulation style of interaction. A mouse, stylus or finger could be used to select visual objects on screen. To turn on a light a user would select the lights icon, select an area of the eHome then finally adjust an intensity bar for the specific light.

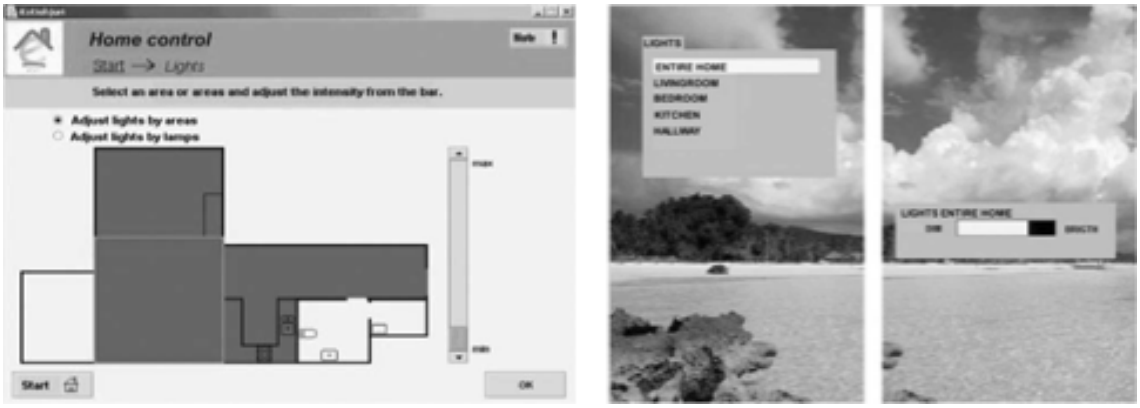

Figure 2-7 - [a] eHome PC Graphical User Interface and [b] eHome Media Terminal User Interface [79]

The PC was also used to setup automatic actions. The media terminal (shown in Figure 2-7 [b]) and mobile phone both used a menu-based method UI (shown in Figure 2-8) whereby the user navigates through a hierarchical set of menus until they arrive at a specific setting, e.g. Lights-->Livingroom-->Turn lights off.

Figure 2-8 - eHome Mobile Phone User Interface [79]

After six months living in the home the users reported liking the controlled automation. The users felt that a PC was well suited for tasks that were not time, location or situation critical such as setting timings or automatic pattern control (e.g. setting the lights to turn on when somebody enters the room).

The users expected to use the media terminal quite a lot, mainly because of their familiarity with television. However, they found the lack of mobility (the IR remote could only be used within the living room) very restricting. Furthermore, the IR *"remote only functioned as a single-user input device, both users could not use simultaneously"* [79]. The mobile phone was by far the most frequently used control device. The users found that it was always within their reach and suitable for instant control. It became their preferred control device because *"unlike the PC or media terminal, they always kept it on"* [79].

Koskela et al.'s study provides evidence for a mobile phone as a preferred interface of smart home control. The traditional feature phone (like that used in the Koskela et al. study) is the predecessor to other forms of personal electronic devices that offer the same mobility, multitasking and high-level of interruptions, but with additional functionality, screen quality and mobility. The personal digital assistant (PDA) is an example of another device that has also been investigated by researchers in relation to smart home control [69, 80, 81]. In a study by Nichols and Myers the PDA is investigated as a "*personal universal controller"* used for controlling smart appliances [69]. The study argues that a PDA interface is superior to the native control interface provided by most household appliances. To test the hypothesis a number of subjects

were asked to perform tasks on a native interface, then on a paper prototype interface and finally on a real PDA. Three metrics were recorded: the time to complete the tasks, number of help requests, and number of missteps. The study found "*that subjects performed significantly better using the PDA interfaces in all three metrics"* [69] and that users of the native interface took about twice as long to complete the tasks, needed external help five times more often and made at least as twice as many mistakes as the users of the PDA interface. It goes on to conclude that *"PDAs, with their rich interface capabilities, are a promising direction to explore for appliance control"* [69].

Although research relating to the mobile feature phone and PDA can be used as evidence in the growing trend towards personal mobile devices for smart home control, they are merely the forerunner to the massive shift in available functionality provided by the smartphone.

The smartphone has recently become one of the most popular forms of computing [85]. In replacing many personal electronic devices it has revolutionised the consumer electronics industry. Although there is no explicit studies gauging the preference of users relating to smartphones (such as the one conducted by Koskela et al. relating to feature phones) their popularity and heritage provide sufficient evidence for their continued use. Moreover, the interest of manufacturers in developing smartphone apps for smart home control provides further evidence for their popularity and potential [85].

Within the available literature, researchers have been investigating the role smartphones have in controlling domestic and office appliances [86, 87], and the abilities of smartphones to be a single device for smart home control [82]. In the case of Suo et al. [82] this interest has manifested in the development of HouseGenie, a universal monitor and controller of smart home networked devices (shown in Figure 2-9). HouseGenie uses an Android or Windows-based smartphone and a unique interface more akin to something found on a desktop computer. This interface consists of a 2D-panoramic view of the entire home with icons for each controllable device. These icons are displayed in the same position in the virtual home as that of the real one. Clicking on an icon enables direct manipulation of the device, for example clicking on the television could bring up a pie-menu with the most common controls.

Figure 2-9 - HouseGenie Smart Home Control Interface [82]

The HouseGenie interface is a novel smart home control system; however, it gives little attention to the user, as demonstrated by the lack of participatory design [82]. The study is more of a showcase for the possibilities provided by smartphones in controlling the home. No effort has been made to involve users when considering the design of the interface or the means by which users control devices. Because it is nearly impossible to design an interface right first time, the interface should be tested prototyped and planned for modification through iterative design [88]. There is also no evidence to suggest the interface is intuitive and easy to use for the user. Despite this, the

HouseGenie system is an excellent example of the potential of smartphones in relation to smart home control.

Many commercial smart home control apps have also been released since the introduction of the smartphone. These apps have mostly been developed by large commercial organisations for consumers to control specific hardware within their home, normally providing an extension to the current form of control already available. Example apps include a television remote released by Philips [89], Sky+ controller [90], Virgin Media TiVo Controller [91], Samsung television remote [92] (shown in Figure 2-10).

These apps could be used to create a semi smart home, whereby each individual app controls a different aspect of the home. Although possible, the fundamental lack of usability across the majority of commercial apps is still a major hindrance for smart home adoption. An excellent example of this is the Samsung remote app. The app, released by one the worlds largest television manufacturers for the Apple iPhone, essentially replicates the functionality of the IR remote control. Using the local Wi-Fi network users can control only a Samsung television.

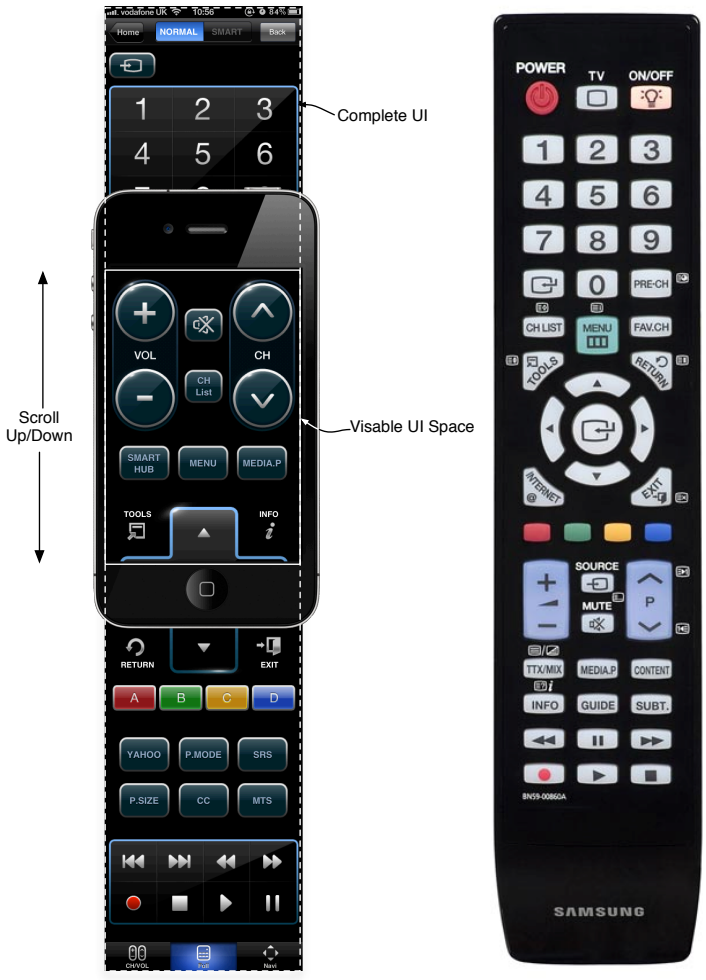

Figure 2-10 - [left] Samsung iPhone Remote and [right] Real Remote

To understand why Samsung designed the iPhone remote UI in such a way, you need to view it next to a real remote (as seen in Figure 2-10). The interface between both forms of control has not changed and there has been no attempt to change the user interface to take advantage of potential UI elements provided by the smartphone (e.g. a QWERTY keyboard for text input). Pan et al. describe how *"TV interface designers should consider consistency with the mobile phone interface"* [93] something that Samsung has seemingly ignored. Moreover, the inability to control televisions from different manufacturers furthers the underlying problems with the current generation of smart control apps. It is this attitude towards design that needs to be addressed if the adoption of smart homes is going to be stimulated. Simply transferring a traditional user interface

to a touch screen smartphone is unlikely to produce the intuitive user experience required

There are, however, a handful of apps that allow user to control a wide range of devices within their home using common smart home standards such as X10 [32] and Insteon [94]. These tend to have a poorer user experience than the commercial apps, normally requiring an intimate understanding of computer networks and protocols. An example of this is the HomeControl app [95]. To operate this app a user must understand the fundamentals of IP addresses and port numbers. They also need to have the unique identifier and protocol of each device being connected within the home.

Figure 2-11 - HomeControl App Interface [95]

Currently these types of apps are only really practical for technophile users. They still fail to address the underlying usability issues surrounding smart home adoption. They do, however, provide a means to control many different devices across a wide range of manufacturers and control protocols.

They could be seen as the starting point for the wider control of the Internet of Things. With each device having a unique identifier, a smartphone app can connect to a wide array of physical devices. With this type of approach, the Internet of Things may

provide a platform for a future smart home. If all of the objects or things within a home were connected to the internet, an aggregation app, similar to that outlined in Figure 2-11, could provide all the control functionality required. However, even if this were the case, the app would still require a basic level of usability. As outlined above, this is not being provided by the current generation of smart home control apps.

Usability is a key issue with the adoption of smart homes. Although there are many different control mechanisms available, the smartphone is key to the future of smart home control. This device has revolutionised how users can interact with their home. Despite this, the same lack of usability has been transferred from the previous generation of control mechanisms to this new device. This is perhaps unsurprising considering the ability to develop for these types of devices has only been possible for the last few years. However, if users are more comfortable using their smartphone for smart home control, then the usability of these 3rd party smart home control apps need to facilitate an intuitive user experience. It is therefore important to understand current tools, processes and support available to smartphone interface designers and user experience architects, for it is these people who will have the largest impact on smart home control interfaces. In understanding the current tools, processes and support available, improvements can be made that may have a profound impact on the smart home control.

2.5 User Interface Design and Prototyping

Interface design falls within the greater area of user experience design (UXD), defined by Unger and Chandler as *"the creation and synchronization of the elements that affect the users' experience…with the intent of influencing their perception and behaviour*" [96]. These elements include a broad number of things including devices a user can touch, hear, and smell. It can also include things beyond the physical domain including digital interfaces or customer service relationships. The user experience of smart home is personal and explicitly linked with an individual's home. Every home is different and therefore the experience of each user will be different. The common link between these experiences is the user interface. By using a common control interface, such as a

smartphone, focus can be drawn on how the user interacts with a given device and how this interaction can be made as simple and efficient as possible.

Brouwer-Janse et al. present an interesting look at the state of user interfaces after the success of the microprocessor [68]. They explain that with the introduction of microprocessor into consumer electronics, the cost of adding new features was dramatically reduced. Mechanical controls meant that adding new features to a product was expensive due to the added material cost. With the microprocessor additional features may only require software enhancements. The cost of software enhancements becomes negligible over large production volumes because the same software is rolled out to every product. This led to an explosion of new features, without the redesign of the interfaces to accommodate for them. Brouwer-Janse et al. also argue that there is a need to *"rethink the interface in terms of the fundamental user concepts and tasks, and not just as an opportunity to complicate the interface with new features"* [68].

As end-users, technical understanding can vary greatly and complex and unintuitive interfaces can considerably impact on the success of a product. A typical user will want to understand how to use a device in the shortest time possible, usually ignoring the manual and opting for a trial and error approach [97]. Therefore, UX designers should attempt to create user interfaces that are as simple and intuitive as possible.

2.5.1 The UXD Workflow

A UX designer should work to a defined workflow of defining, designing and developing what an application, including its interface, will do and how it is best achieved with the budget and timeline available [96]. Various techniques can be deployed during each phase; for example, wireframing and prototyping for the designing phase will improve the end result.

2.5.2 Requirements

The beginning of any new software project normally starts with the gathering of ideas related to the features of the system and the expectations of the stakeholders. These ideas are the basis for the system and are refined into clear and detailed requirements

using additional information gathered from the stakeholders. This gathering process is important and although this thesis is ultimately concerned with the design, prototyping and evaluation of user interfaces, having a clear understanding of the requirement gathering and analysis process is essential in understanding the later phases of the UXD workflow.

The early stages of a project will generally produce a set of *fuzzy* requirements. These are requirements, which, although informative, do not give enough definition to be full requirements. Gathering all the information from the stakeholder is essential in clarifying these user requirements. Having a detailed understanding of the current state of any previous work that would likely be integrated into the proposed solution is also important. Using these user requirements a project requirements list can be generated with prioritisation weightings attached to each requirement thereby creating a focus for each stage of the project. This process has been illustrated in Figure 2-12.

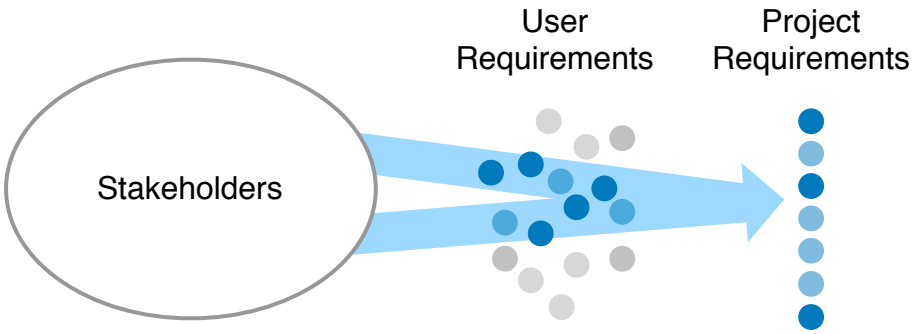

Figure 2-12 - Producing Project Requirements using Information from Stakeholders [96]

Some of this information will come from the stakeholders during various meetings and conversations; other information will come from user-focused research.

2.5.3 Designing

Figure 2-13 shows the high-level, standard user experience design flow for any application or software project. Initially a concept design is used to understand the application in its entirety and this feeds into task flows and wireframing. Task flows

help to identify how user tasks will flow into other tasks based on decision points in the process. These flows can then be used to produce a wireframe of the structure of the application; *"a low fidelity prototype of an…application screen, a wireframe is used to identify the element that will be displayed on the…screen* [96].

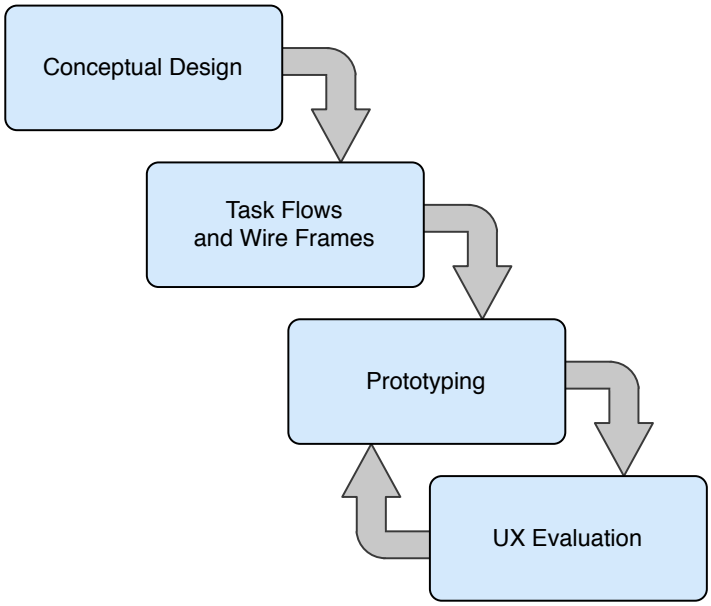

Figure 2-13 - User Experience High-Level Design Flow [88, 96]

2.5.4 Prototyping

Prototyping happens within the analysis and design phase of the larger system development life cycle. It is the "*process of building a model of a system and, in terms of an information system, prototypes are employed to help system designers build a system that's intuitive and easy to manipulate for end-users" [98]*

During the analysis phase, system analysts gather information from the stakeholders regarding the user requirements (shown Figure 2-12). Prototyping can augment this process by converting these basic and sometimes intangible specifications, into a tangible, but limited, working model of the desired system [98]. The addition of a tangible model enables user feedback to be gathered from a physical system, which users can touch and see. Furthermore, the evaluation of the experience between a user

and a prototype, allows the system analysts to modify existing requirements, as well as develop new ones, based on the user feedback.

Typically once a design is complete it is tested with users and the problems the users faced are noted. These problems would then be fixed in a new iteration, which would again be tested with users. This ensures that the fixes did indeed solve the problem and it attempts to find any new usability problems still outstanding or introduced by the changed design. *"Iterative development of user interfaces involves steady refinement of the design based on user testing and other evaluation methods"* [99]. It is this iterative design model that will be the focus of this thesis. This is based on the model being the most established, cheapest (it is often possible to iterate within a few hours) and strongest (the model will keep going for as many iterations as a budget allows) [100].

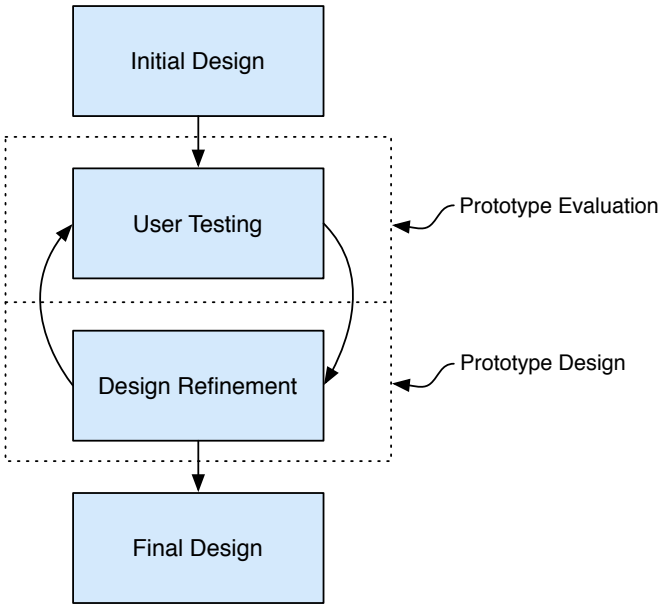

Figure 2-14 - Iterative User Interface Design Process

One of the most effective ways to improve interface design through prototyping is to use the iterative prototyping evaluation process. As shown in Figure 2-14 this process can be split into two majors areas, the design of the prototype and the evaluation of the prototype. Within the UXD workflow this process offers one of the best ways to

investigate possible improvements, thereby enabling superior user interfaces for smart home control. This is because these stages actually involve real world users, giving designers real world feedback about their design. Previous experience and pre-defined standards can only enable conceptual designs to be created, wireframes and possibly first generation prototypes. Unless some iterations of a prototype have been tested with real users the design will most likely have poor user experience [99]. The correlation between design iterations and interface quality can be seen in Figure 2-15.

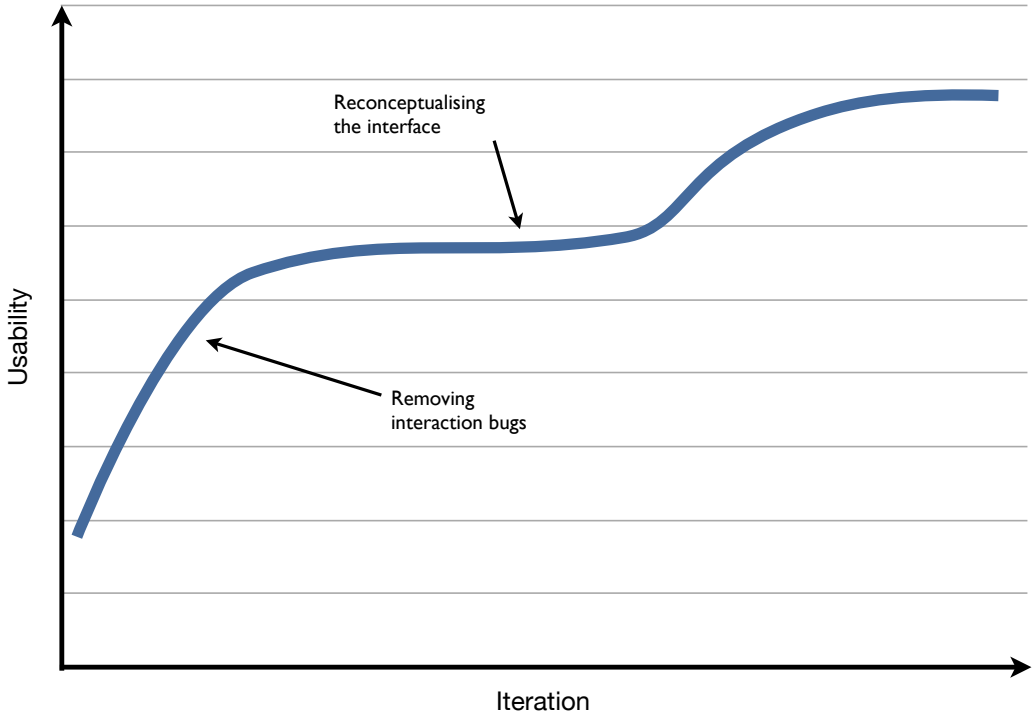

Figure 2-15 - Interface Quality as a Function of the Number of Design Iterations [99]

Figure 2-15 shows iterations plotted against usability. As more design iterations are completed the more usable the interface becomes. Initially this is due to the correction of interaction bugs (e.g. common errors introduced during the design process). The usability is then improved via a reconceptualisation of the interface. This would include fundamental design changes based on feedback and observations from the user [99].

In general, prototyping can provide designers with quantifiable user feedback whilst at the same time reducing development time and cost [98]. It has been used widely for over two decades and is arguably one of the best processes for improving interfaces [101, 102].

However, with the introduction of smartphones, and the revolutionary touch screens they incorporate, many of these traditional tools simply do not lend themselves to this new form of interaction. Furthermore, any techniques that use a traditional keyboard and mouse, or any form of input from older generation phones, simply cannot replicate the same input mechanisms available to newer smartphones such as multi-touch gestures.

2.5.5 Prototyping for Mobile Devices

The previous generations of mobile phones, now classed as feature phones (outlined in section 2.4.4), were only capable of providing simplistic functionality usually in the form of multiple menus. The market leader of feature phones was the Finnish company, Nokia. Lindholm et al. [103] present an interesting study on the usability of Nokia mobile phones. In the study Lindholm et al. argue that the mobile user interface should be focused on user needs, *"knowing the user allows us [Nokia] to select the most relevant features for most situations, which in turn enables us to present compact and portable designs" [103]*. Using this theory mobile user interfaces can prioritise which function will be most relevant to the user and present them in a familiar manner, in Nokia's case a sequential menu. This is an important consideration for mobile UXD especially when applied to the control of smart homes or smart devices as user priorities can be constantly changing based on things such as location, time of day and activity.

Lindholm et al. also state that they *"believe that the user interface should be developed through evolution, not revolution"*, something that enabled Nokia to constantly refine its user interface. Although this belief is clearly one that has been applied to the development of existing mobile user interfaces, it fails when there is a major change in the underlying phone hardware. This issue is strikingly apparent when you consider the current fate of Nokia and its Symbian operating system (OS) [104]. With the movement

towards smartphones there has been a massive shift toward Apple's iOS and Google's Android operating systems. Both the these operating systems came to prominence after a revolution in mobile user interface design due to the use and availability of large colour touch screens. Nokia has been unable to evolve its current OS and has instead chosen to decommission it in favour of a more revolutionary OS developed by Microsoft.

Nokia's situation shows that UX designers need to focus their efforts on tools that allow them to build for this new type of interface. Simply using the same techniques as those developed for feature phones are not sufficient for the new generation of smartphones. Furthermore, the traditional tools are physically incapable of providing support for gestures – one of the major usability differences between feature phones and smartphones.

Gestures are a pre-defined physical actions performed by the user to enable intuitive control. A selection of gestures can be seen in Figure 2-16. A classic example is the spread/pinch gesture. Using this action, a user can, for example, mimic a zoom control. Spreading the finger apart will zoom in an image in while pinching them together will zoom it out.

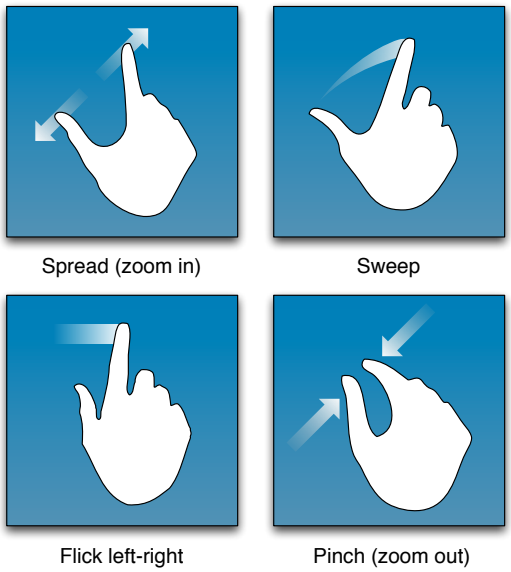

Figure 2-16 - Example Gestures [105]

Gestures enable a fundamentally different user experience compared to feature phones. Before their introduction it was very difficult to have an intuitive user experience on a mobile touch screen device [106]. This was because the user had to adapt to a rigid set of menus rather than a variable touch screen interface. Because of the hardware and software requirements required for this type interaction traditional prototyping tools simply cannot replicate the experience of a touch screen device capable of recognising gestures.

Traditionally, research into mobile device interface design was mainly conducted by the large mobile phone companies such as Nokia. The introduction of the mobile phone in the mid 2000s was characterised by large phone manufacturers offering monochrome displays and a standard set of pre-installed programs. The inability to freely design and distribute software for the majority of mobile phones meant that there was no interest in the field of mobile phone interface design by anyone other than the phone manufacturers. Even if a 3rd party company had the ability to install software, the rigid interface, consisting mainly of menus, meant there was limited scope for any real interface design.

Now, however, with the ability of any 3rd party to develop and distribute apps for smartphones, twinned with the introduction of gestures, there has recently been an increased interest into the development of a new generation of prototyping tools for smartphone interfaces.

Many commercial solutions have begun to appear on the market aimed at a wide audience and the production of low cost, low fidelity prototypes [107-113]. The interest can be attributed to the introduction of dynamic, full colour, touch screen displays and the ability to execute 3rd party apps on a mobile device.

2.5.6 Existing Toolkits For Rapid Prototyping Of Interfaces

The recent developments in smartphone prototyping particularly target the Apple iPhone and the production of low fidelity prototypes. These low fidelity solutions include paper prototyping kits such as Notepod and UI Stencils [108, 109] that enable faster physical drawing of iPhone GUI elements. These extremely low-fi solutions are

designed to be used without the aid of computer and are akin to a *back of a napkin* type technique. Initial concept designs or wireframes can easily be mocked-up using these types of tools but they are extremely limiting when the prototypes are tested with real users. This is due to the fidelity being so low that the conceptual leap required by the user, in realising the end interface is extremely large. Other drawing libraries such as MetaSpark's Fireworks vector [110], iOS GUI PSD [114] and Graffletopia Stencil Kit for Omnigraffle [112] are still for low fidelity paper prototypes (or app mock-ups), but they facilitate computer aided drawing of an app, which is then printed and tested with users. Examples of the interface elements included within Graffletopia Stencil Kit can been seen in Figure 2-17 [b]. Using these interfaces elements, the basic components of an interface can easily be included within a low-fi prototype.

Figure 2-17 [a] shows a number of Notepod pads. These are similar to Post-it Notes with the concept being the use of an individual note to hand draw the basic interface components for each app screen. Figure 2-17 [b] shows Graffletopia Stencil Kit, which is used in the same way as Notepod pads; however, it digitises the process enabling components to be used from a library rather than individually drawn.

Figure 2-17 -[a] Notepod [108] and [b] Graffletopia Stencil Kit [112]

Bolchini et al. [115] demonstrate alternative attempts at increasing the realism, or fidelity of smartphone interface prototypes. Using low fidelity paper prototypes, a simple drawing of the each application *screen* is generated. These are then digitised and displayed on a device using the built in photo library application. Users can then view a proposed interface and jump between different *screens* in the same way they would jump between photos. This semi-interactive prototype is a novel idea that can be produced and deployed relatively easily. However, being based on low fidelity graphics means the overall realism of the prototype is still poor and the interaction very limited. This basic process can be seen in Figure 2-18.

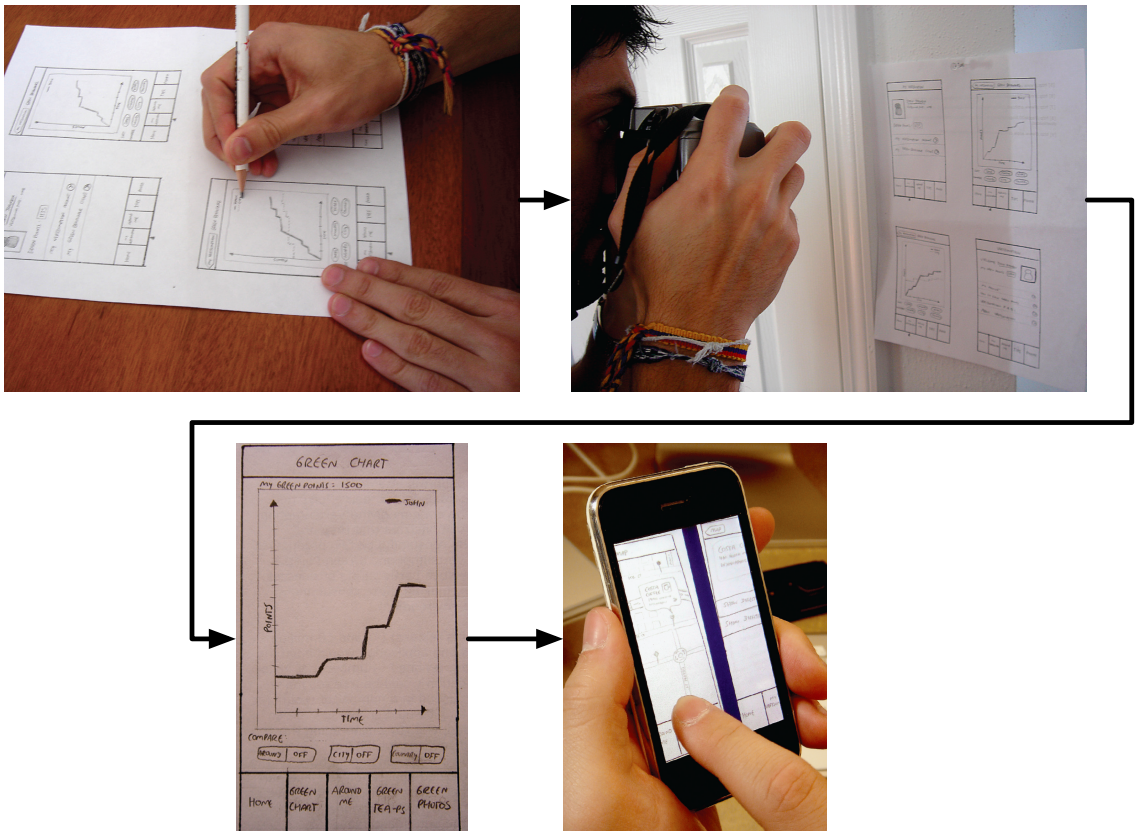

Figure 2-18 - Prototyping iPhone App Using Technique Developed by Bolchini et al. [115]

Low fidelity prototypes offer quick and easy solutions to prototyping and these examples are by no means an exhaustive list. The results, however, are extremely

limited. The conceptual leap required when looking at a simple pencil drawing is both difficult and unnecessary. The complete lack of functionality also severely hinders the usefulness of such prototypes. As demonstrated by Hennipman et al. [116], Houde et al. [117] and Carter et al. [102] prototypes should be high resolution "*in order to let people concretely visualize the design"* [117].

Meskens et al. [118] present a system for the creation of cross platform prototypes. Using different XML renders, the system works on the principle of a build-once-fit-all paradigm, similar to web technologies. Although this paradigm can be considered convenient there are many issues with the system that limits its use. At the core of the problem is its complete omission of the differences between different devices. Designing for a small mobile device is fundamentally different compared to desktop applications. Screen size, gestures, button size, etc. all need to be considered as part of a good user experience. The tool fails to incorporate any modern form of interaction (such as gestures) and it renders elements in a similar way across all devices. Even after two years of development [119] an updated version of the tool still failed to incorporate any of the requirements for touch screen and gestures (see Figure 2-19).

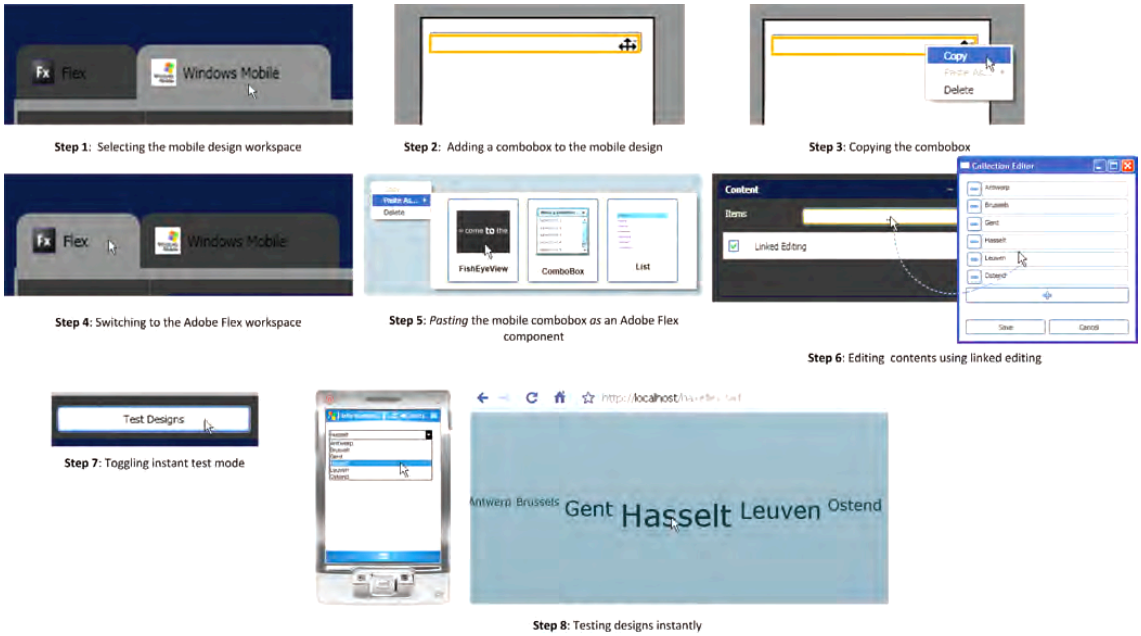

Figure 2-19 - GUI Element as Part of Jelly: a Multi-device Design Environment for Managing Consistency Across Devices [119]

Examples of commercial solutions with increased fidelity show more promise. Blueprint [120] and Keynotopia [121] are solutions that offer the ability to use high-resolution computer generated graphics, but go further in functionality by allowing links to connect *screens* together. Links can be applied to different graphical screen elements so that when a user taps an element a new screen is loaded (shown in Figure 2-20).

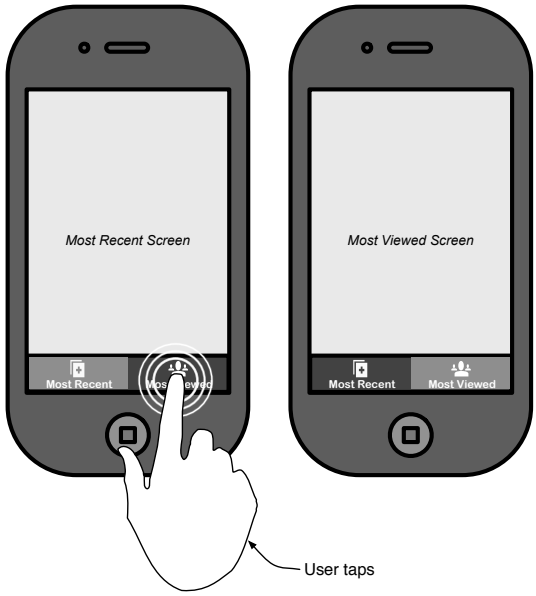

Figure 2-20 - Example Prototype App with Screen Linkage

This is a significant improvement on the techniques demonstrated by Bolchini et al. as it provides realistic interaction. Users can navigate between different screens in a similar fashion as the final app. The visual fidelity of the graphics can also be high. However, they both still fail to incorporate the main difference between a traditional user interface and touch screen interaction: gestures. Simply tapping on graphical elements does not give the same user experience as that provided by a real touch screen device. Gestures are one of the most important differences between the previous generation of feature phones and PDAs and todays smartphones. Omitting this type of control from a prototype interface instantly reduces the realism and accuracy of the data gathered during user centred testing. It also limits the design of interfaces, as user experience designers may be less likely to consider their use based on lack of user testing. As the

existing smartphone interface design tools do not incorporate gesture functionality this prevents them from representing a smart phone interface paradigm.

The current prototyping solutions also severely hinder the iterative prototyping process. Deploying prototypes can be time consuming and difficult. In the case of Keynotopia, the need to plug the device into a computer to transfer the relevant files is cumbersome. Realistically, this also limits the deployment of the prototype to anywhere within a small physical distance of the original design. Blueprint requires that the prototype be produced on the device using a bespoke app, rather than a desktop computer. This means the designer has to have an intimate understanding of the graphics tools available within the specific prototyping app. Furthermore, it also requires that any graphical assets must be reproduced in a desktop application for final integration into the production app.

In early 2012 Adobe released Proto, an app that *"lets you create interactive wireframes and prototypes of websites and mobile apps on your tablet. Communicate and share ideas with teams and clients using a touch-based interface"* [122]. It essentially digitises the drawing process whilst adding the element of interaction into the prototype, similar to that employed by Blueprint Viewer or Keynotopia. Unlike Blueprint Viewer or Keynotopia the prototypes are merely interactive wireframes developed on a touch-based interface (e.g. iPad).

Figure 2-21 - Example Adobe Proto Prototype [122]

As shown in Figure 2-21 the fidelity of these prototypes is extremely low. It could be argued that this low fidelity increases the speed in which the prototypes are developed. However, the low fidelity provides little understanding to the final design and acts as an aid for determining the architecture of the app rather than a high fidelity design orientated prototype. It is also heavily geared towards web design rather than mobile app development. The reliance on a touch-based interface for prototype development, appears to have also limited the appeal of Proto, and by the end of 2012 it discontinued support for the product, citing a lack of overall success with many of its touch-based apps [123]. However, with the entrance of Adobe in the market, interface prototyping can be seen as *hitting the mainstream*. It is no longer a process that might be completed if the time and budget was available; rather, it is becoming a necessary step that needs to be completed if a product or app is going to compete in an increasingly competitive marketplace. Despite this, the limitations with the current tools available for rapid prototyping of smartphone interfaces dramatically affect the usability and quality of

smart home control interfaces[2]. Within the current literature the abilities of any system in providing the type of high fidelity, gesture enabled, rapid prototype solution is limited.

Within the user experience design cycle, the process of iterative prototyping is best placed to offer improvements to the interface and usability of smart homes [100]. Based on the preference of the smartphone for smart home control, improvements should be focused on the prototyping techniques and tools available for this type of device.

The limitations of the current tools for smartphone prototyping, especially the focus on low fidelity prototypes, is major factor for smart home usability, especially when considering Hennipman et al. found that "*it is important the 'front end', the part of the prototype visible by the end-user…be as real as possible*" [116]. The development of quality tools that enable rapid high fidelity prototypes to be produced, which also take advantage of the current interface paradigms implemented by a modern smart phone, would present a novel and interesting area of research. With these tools the iterative design process would be facilitated and an improvement to smart home control interfaces could be promoted.

However, simply improving prototyping tools for the device will only address part of the iterative user centred prototyping process outlined in Figure 2-14. Observing and evaluating a prototype being used by a user and feeding the results back into design completes the loop. This could be achieved out of the smart home context but characteristically that is unlikely to provide a complete solution because context affects interaction. To fully understand how the prototypes will be used it needs to be observed in a smart environment.

2.6 Evaluation: Observation, Capture and Simulation

As part of any prototype evaluation exercise, gathering quality, quantifiable user feedback is paramount. This feedback can be gathered in a number of ways, but usually

[2] The recent abilities of Axure are discussed in section 8.2.2

the main methods involve some form of empirical techniques. This can be through simple and casual observation with hand written notes, to more formal techniques such as standards compliance. Nielsen [124] outlines a number of usability inspection methods: *heuristic evaluation, cognitive walkthroughs, formal usability inspections, pluralistic walkthroughs, feature inspection, consistency inspection and standard inspection*, all of which involve empirical analysis. One method for capturing the raw data produced through this type of analysis is through audio-visual based recording techniques. These types of capture mechanism are used in all the research-based, real world, smart homes presented in this thesis [4-6] (discussed section 2.2). It is also employed in a number of pervasive computing evaluation spaces [125, 126], discussed in section 2.6.4.

Using a large array of audio and video sensors offers a number of advantages including the *"recording of the entire context of an interaction (e.g. use of documentation, environmental distractions, etc.), the ability to hear verbal or thinking-aloud style comments made by the user, and simply the ability to see exactly what the user is doing at a given point"* [127].

Audio-visual recording techniques are especially important due to the large amounts of information that can be extracted from the data stream. The availability of this high quality information has led to the wide use of audio-visual recording for evaluating empirical-based analysis techniques [127].

As well as audio-visual data recording other, more quantitative forms of data capture, can be gathered to offer a greater level of detail. One example of this is computer logging. This type of data capture provides an unobtrusive means of capturing low-level *semantic-free* data specific to an individual action or task. Logging has the advantage that it tends to be a cheap method of gathering large volumes of data. However, the disadvantage of generating these large datasets is that the data can easily become unmanageable and the lack of *semantics* means that it is nearly impossible to analyse why an action was performed or how it was structured [128].

On a more basic level, user notebooks can be used so that participants themselves can keep logs of activities or problems. This would be useful for extended user testing

operating over periods of hours, days or months. Having an iterative prototyping process that could potentially span months or years is, in this instance, both impractical and unrealistic, however similar participant based techniques may have more relevance. Interviews and questionnaires are often used as they allow a structured way of gathering information. Interviews have the advantage that questions can be quickly tailored and varied to suit the context and level required. Furthermore, the interviewer can probe more deeply on interesting issues following a top-down approach (starting with a general question about a task and progressing to more leading questions) [128]. Interviewing falls within the category of data gathering that can also be captured using audio-visual recording techniques.

In reality a mixture of recording methods will give better results, but focusing the majority of capture around audio-visual recording techniques will often give the best all round coverage [127]. Using different configurations can also further enhance the quality of the recordings, e.g. a separate audio recording can be used with superior microphones or the use of stereo audio recording can help locate off screen noises [128]. Audio-visual based capture can also be synchronised with logged data to enable the extraction of extremely detailed information, with addition of semantic information, relating to a specific task.

Testing with users, however, is not as simple as leaving users in an empty room. In order to provide a realistic experience the user should be located in an environment akin to a smart home. The intimate nature between a person, their home and the device they are using, requires a review of the tools available for producing such an environment.

One way of producing a smart environment to test prototypes, is to build a smart home. This has been accomplished by many different research groups, as outlined in section 2.2, and the resulting environments have been used for various user centred experiments. However, there are many issues with using a physical smart home for the evaluation of prototypes.

First, they are physical. It is unrealistic to consider moving such a home or easily producing a duplicate. They are prohibitively expensive to the point where they are only really developed by large research or commercial institutions. All of the major smart

homes outlined in section 2.2 are custom designed, bespoke developments that have been built from the ground upwards. They are also generally very difficult to upgrade. New devices require large development efforts to successfully integrate them into the pre-existing home. The average user may also have none or limited experience with a new environment such as a smart home. Expecting a UX designer to understand how such an environment works on a technical level is unrealistic.

These types of issues present such a high access bar that it is virtually impossible to empower and equip the usability community with the tools, and relevant research to have a major impact on smart home usability.

Due to limitations of building a genuine and functioning smart home, simulation is being proposed as an alternative to such a development. As will become evident, simulation has been used in many research projects aimed at reducing the overheads associated with producing a real smart home for the purpose of conducting research.

Kheir describes simulation as *"the process by which understanding of the behaviour of an already existent (or to be constructed) physical system is obtained by observing the behaviour of a model representing the system"* [129]. Thus simulation is the development and construction of a model, which through an analytical process can be used to study a problem. A model can be a representation of an object, system or idea and through the study of the mechanisms of the object, system or idea, such a model can be produced. Lehman states that "*simulation…is itself a process – the operation of a model – but a process that is in some sense a copy of or parallel to a real process"* [130].

Although simulations are often conducted on a computer, the definition should not be restricted to this medium. The game of Monopoly, for example, is really a simulation. The physical game (board, dice, houses, etc.) is the model and the game play is the simulation [130]. The model represents a real world system, the property market. The operation of that model, playing the game and buying and selling of property, is the simulation. Obviously Monopoly is not an accurate simulation of a real world system and this can be measured with the *validity* of the model [131]. The more closely the model represents the real system the more valid it becomes, thus making it more

effective. If the game of Monopoly had a more valid model it would increase the accuracy of the simulation.

Models and simulations are either virtual or physical. For example, a building can be designed and modelled using a computer and thus it is a virtual model. This virtual building can be placed in a virtual wind tunnel and the effects of air flow over the building simulated. Alternatively, a physical model of the building can be placed in a physical wind tunnel and the effects of airflow simulated. Both virtual and physical models are still representations of reality; however, the *validity* of the model may vary greatly. The same can also be said of simulations. This blurring of the physical and virtual world gives designer's different options to consider as each has different benefits.

Simulation models provide outputs of a system for a given set of inputs. All simulations are therefore input-output models, which are *run* rather than *solved* to produce the desired information or results [132]. Simulation models are therefore unable to produce results on their own, *"they can only serve as a tool for the analysis of the behaviour of a system under conditions specified by the experimenter"* [132]. Simulations produce valuable information, which can have a number of profound benefits for the production of an idea, system or object. It can be used to check and optimise a design before its construction, thus helping to avoid costly design errors.

Virtual simulation offers benefits over physical simulation. Although the initial development costs may be large it can often be cheaper to run many simulations. The cost of executing computer code is usually significantly cheaper compared with building multiple physical simulations. This means many virtual simulations can be executed with minor adjustments, enabling the simulation model to be refined. The major limitation of a virtual simulation is the inability to interact with any form of tangible object.

Alternatively physical simulation provides the user with a physical, tangible object with which they can interact. This is very important for conducting simulations that require a tangible form of interface that may need to be touched or held. In the case of smart home control with a smartphone, this is extremely important. A smartphone is a

physical object that has a fundamentally different interface from the traditional keyboard and mouse and it needs to be physically felt and interacted with. Ultimately some form of simulation that can merge both the physical and virtual simulation environments, will offer a superior simulation model.

2.6.1 Simulation In The Development Cycle

Within a product development cycle the use of simulation and modelling is used as it allows the identification of problems, bottlenecks and design shortfalls [133]. Traditionally the product development cycle would begin with a set of requirements that are analysed and used for the construction of the product. After testing the product, or system, it is complete. A traditional approach not only omits the need for simulation it also relies on the initial requirements being correct and rigid before the development cycle begins. Altering the requirements once the development cycle has commenced increases the cost and effort required to realise the final system.

Including a simulation phase (as shown in Figure 2-22) would facilitate the overall quality of a product while also reducing the need for correct and rigid pre-defined requirements. This is because the modifications to the requirements can be made based on analysis and results from the simulation phase without the large overhead of actually building the system.

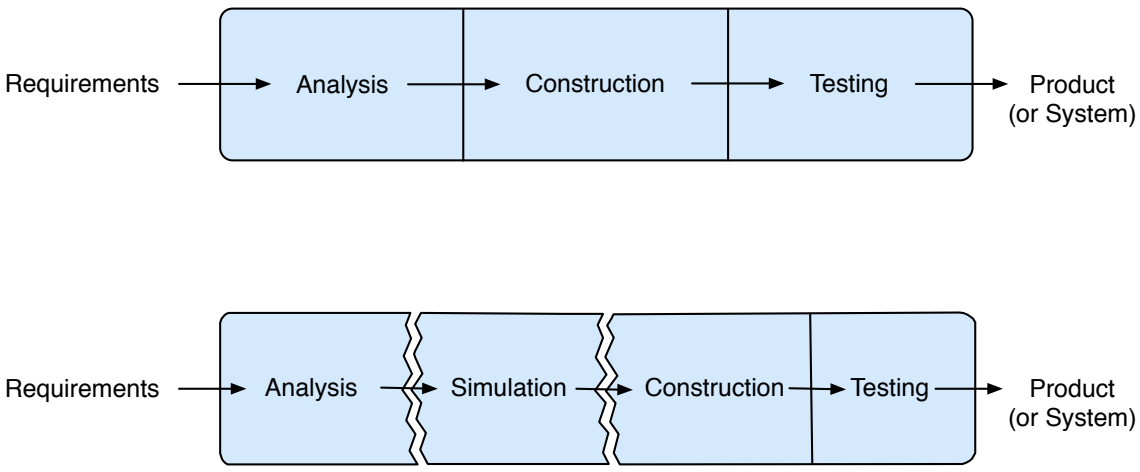

Figure 2-22 - Traditional Development Process with/without Simulation [134]

2.6.2 Simulation as the End Goal

Thus far simulation has been discussed as an additional phase within the product development process. As seen in Figure 2-22 this occurs before the construction phase providing key information, which can lead to an improved product. However, having a process with the end goal being a product or system is not the only use for simulations. In some cases simulation can be a replacement for a physical construction or system. This, of course, requires the elimination of an end product and the reassignment of the end goal being the simulation and the extraction of data from that simulation.

This may seem to counter the previously defined notion of simulation, but when considered within a larger process involving the development of many systems, the benefit of simulation as an end goal becomes apparent.

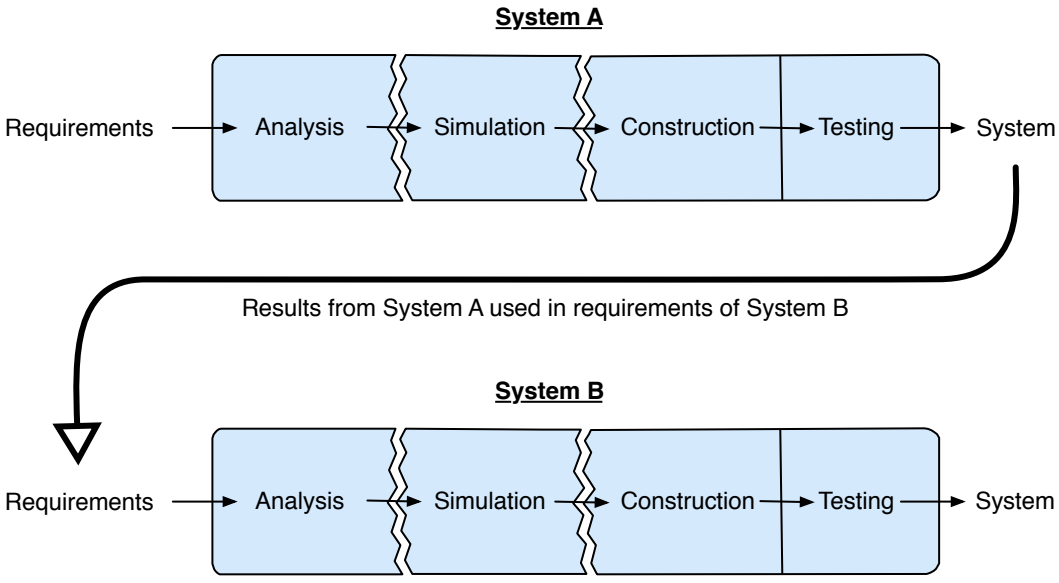

Figure 2-23 - Multiple System Development Process with Simulation

Figure 2-23 shows the development of two independent systems with the inclusion of a simulation phase. The key point being made is that the sole reason for the development of one of the systems, *System A,* has been to enable the generation of the requirements for *System B.* If there was no physical need for the construction of *System A* and its

construction was merely a means of gaining the information required for *System B* then there may not be a need for its construction in the first place. If a simulation of *System A* produced the required information then the end of goal of *System A* would not be a final system, it would be a simulation. This can be seen in Figure 2-24.

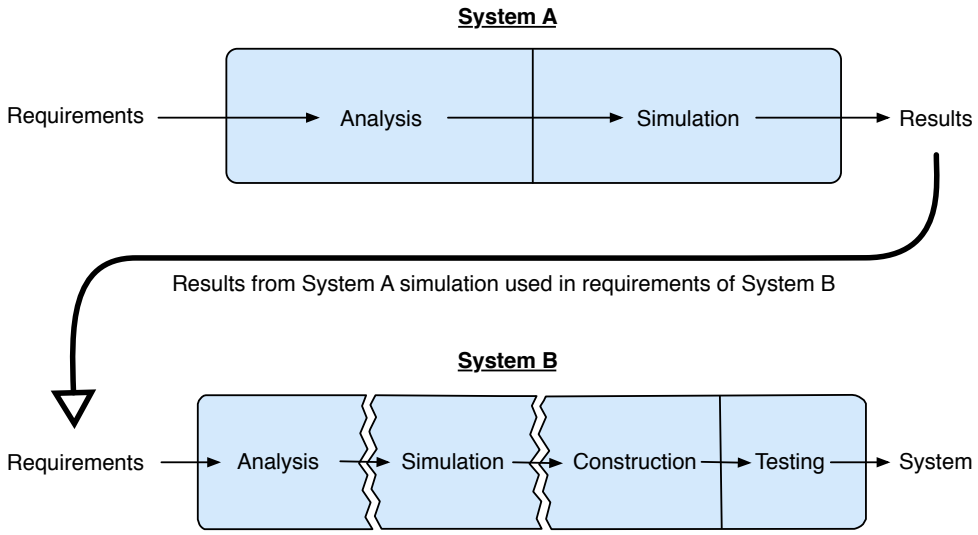

Figure 2-24 - Using Simulation to Gain Information for Different System Development

The benefits of not constructing *System A* could be enormous considering the majority of system development effort occurs after the analysis and simulation phases. This can be seen in Figure 2-25 where the combined total of the effort consumed on construction, system testing and release is 70% of the total consumed throughout the entire project. Compare that with the 30% on requirements, architecture and design and the potential savings possible by not building the end system become strikingly apparent.

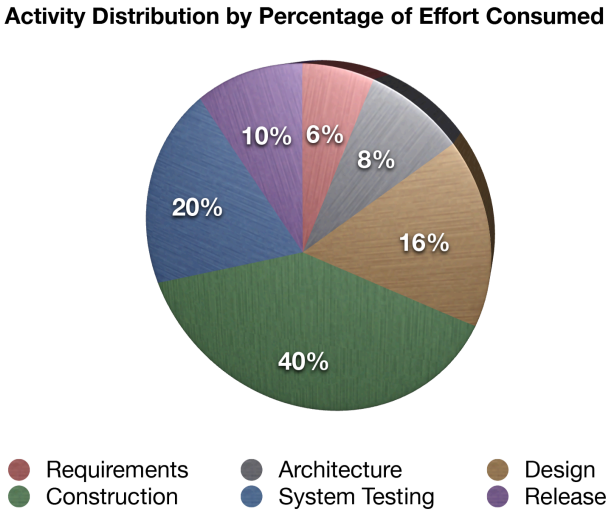

Figure 2-25 - System Development Activity Distribution by Percentage of Effort Consumed [135]

However, to achieve this the *System A* simulation would require an extremely high validity in order to obtain accurate data. The increased complexity and cost of such a simulation may, in some cases, outweigh the savings gained from omitting a complete construction.

In the context of smart homes, however, simulation would offer a valid platform for reducing costs and complexity if such a home was being used to produce results for a different system. If we consider the current generation of research focused smart homes there is the potential to replace these homes with a simulation without a severe knock-on effect to research results. Prototyping and evaluation of such interfaces would, of course, require a smart home environment, but this environment may not need to be a physical construction (like the current generation of research smart homes). In this instance the sole purpose of the smart home is to provide information for another system; the development of user interfaces, and for this a physical simulation is all that is required.

Similar ideas of simulation, in relation to smart homes, have already been explored through the use of virtual simulation. Lertlakkhanakul et al. present a study [136] investigating *"how to create and to implement virtual space using virtual reality technology as a platform to simulate smart home service configuration"* [136]. The

simulation produced is similar to a computer game and it allows for a collaborative effort between users and designers in deciding the optimal configuration on a smart home. Specifically the virtual simulation model is capable of specifying actions, areas and objects for different purposes [136].

Figure 2-26 - V-PlaceSims Screenshot [136]

Although this type of simulation offers many benefits (such as cost and speed of development), it has a severe limitations with regards to tangible interfaces. Figure 2-26 shows the implemented virtual environment. The manipulation of this environment is through the use of a desktop computer using a mouse and keyboard. This type of simulation is therefore unable to support a tangible device such as a smartphone.

Armac et al. also present a smart home simulator called the *eHomeSimulator [137]* [138]. This virtual simulation tool enables the simulation of different smart home environments allowing a developer to *"abstract from creating buildings and developing or purchasing devices"* [138] and the associated high effort and financial costs. This

disconnect from a financial burden empowers developers and designers to create many new environments using arbitrary architectures and devices. Due to this, the eHomeSimulator can be used to develop new and radical types of smart home services with greater evaluation intensity on the specification, configuration, and deployment processes.

Using three main simulation elements: an environment, a device and an agent, a virtual smart home can be created and manipulated in real-time. Various scenarios and services can be played out in the simulated virtual environment including location-based services (e.g. music can follow an agent as they move between rooms). A front end GUI presents the user with a game like interface showing an agent located within a larger virtual environment. Using a control panel the user can control how the agent moves and interacts with the various devices located within the environment. This can be seen in Figure 2-27.

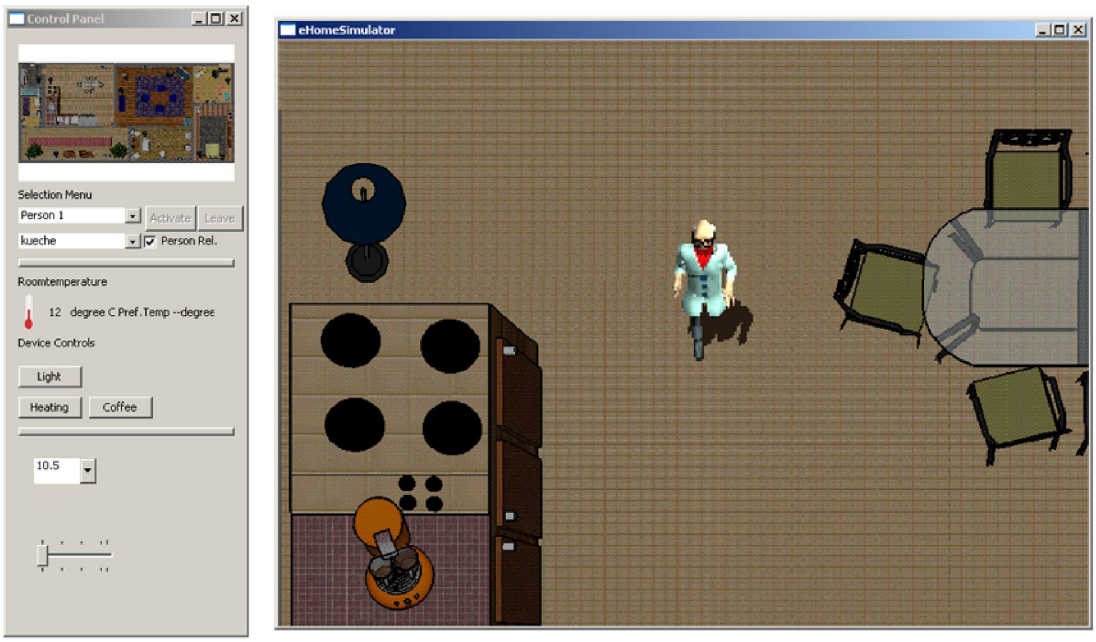

Figure 2-27 - eHomeSimulator GUI

The eHomeSimulator is an excellent example of how simulation can be used instead of construction to reduce the cost, effort and complexity of a system without a reduction in quality of the result produced (assuming the validity of the simulation is high). Where the eHomeSimulator would fail to produce reliable results is with any scenario that required a high-level of interaction. The intimate way a person interacts with their home is extremely difficult to simulate using a virtual environment based simulation. A desktop simulation simply cannot currently replicate the immense details and cognitive challenges faced by a user when interacting with a real home. This is outlined by Picard, who discusses the term *affective computing* as any computing that relates to, arises from, or influences emotions [139]. It is this lack of support for affective computing that hinders a desktop simulation such as the eHomeSimulator. Furthermore, the lack of any tangible control interface means the user has to interact with the simulation in a fundamentally different way to how they would in a real world environment. This limitation is extremely difficult to overcome using any virtual representation of the real world. Due to this a different approach to smart home simulation is required. The main requirement for this different approach is an environment, which fuses together the real and virtual worlds enabling cost efficient simulations to be developed and deployed, not just on a computer screen, but rather, a physical space where a user can interact with a tangible control interface and virtually powered simulated devices.

With the use of simulated devices, a means of control is required. With the physical research smart homes discussed in section 2.2, control of devices came from either user interaction or through context aware systems. The system level logic was actually implemented so that real automation or control could be used. The same system level logic could be applied to a simulation, but as outlined in section 2.2 this is both financially costly and time consuming. It also produces rigid systems that are difficult to change or adapt to various forms of control and interaction. The Wizard of Oz technique is therefore being introduced as an alternative.

2.6.3 Wizard of Oz

The Wizard of Oz technique involves studies where a user is told they are interacting with a computer system though they are in fact interacting with a human operator, or *wizard* [140]. The term has been coined, in part, after the popular children's book, The Wonderful Wizard of Oz [141]. An easy analogy is to think of the classic Hollywood film [142]. In the film Dorothy and friends are led to believe the Wizard of Oz is all great, powerful and magical. However, when the curtain is pulled back it is just an old man pulling levers and speaking into a microphone. Using this analogy, if the computer system is considered to be the wonderful magic, it can easily be replaced with a wizard whose responses are similar to the old man pulling levers, but in our case the wizard is driving the simulation. With this technique *"the subject can be given more freedom of expression, or be constrained in more systematic ways"* [140].

Wizard of Oz studies have traditionally been the domain of natural language processing research [143]; however, it also has an important role in prototyping graphical user interfaces. Molin [144] investigates the use of the Wizard of Oz technique for exactly this purpose and found it useful for the improvement of the interactive design process. Molin's experiment involved the GUI design for the control of a robot arm. This arm was used to guide a surgeon with the insertion of hip fixation devices and the GUI consisted of a touch screen combined with a trackball and mouse button. However, instead of implementing the system, which connects the user interface to the control mechanism, a wizard was put in place to mediate the commands. Various GUI designs were tested during twelve user sessions and results were recorded relating to both the use of the Wizard of Oz technique and GUI design. Molin found that the Wizard of Oz prototyping allowed for *"a true interactive experience without [the need for] traditional programming"* [144]. The time taken to implement new ideas was dependent on their nature, but the design sessions revealed that some changes, which traditionally would require a major rebuild of a programmed prototype, could be implemented in a few minutes. Importantly, Molin found that Wizard of Oz prototyping can *"produce different types of output to support and extend the requirements specification. The prototype itself can be used by designers and programmers. Screen and video*

recordings of sessions can be used as clarification and examples of good or bad design" [144].

What is required, therefore, is a similar simulation environment as outlined by Molin. However, instead of being focused on medical equipment the focus is shifted towards a smart environment. Two similar environments have been produced before called EasyLiving and Cooltown. Microsoft's EasyLiving [145] and Cooltown [146], from Hewlett-Packard Labs, are physical pervasive environments that aim to aggregate diverse devices into a single unified user experience. These types of environments will usually exhibit context aware abilities and as such they are often referred to as intelligent environments. One example of this type of intelligence would be the environment's ability to learn the user's preferred lighting arrangement when watching television. These physical intelligent environments have been produced with the aim of allowing the development and evaluation of ubiquitous computing devices within a specified context.

2.6.4 EasyLiving

The goal of the EasyLiving project was *"to develop a prototype architecture and technologies for building intelligent environments that facilitate the unencumbered interaction of people with other people, with computers and with devices"* [145]. These devices, computers and people came together in a physical space to provide access to information and services. Devices and computers could be static, such as ceiling lights or large televisions, or mobile, such as laptop computers or mobile phones. The variety of devices ranged from traditional input and output devices such as mice, keyboard, speakers and displays, to more novel ones such as an active badge RFID system, cameras, home entertainment systems and wall-mounted displays [147].

The physical space where these heterogeneous elements were brought together was the EasyLiving house. Here extensive testing was conducted and the premise of the automated intelligent environment could be evaluated. As part of this evaluation process four key design issues were prioritised: sensing, user interfaces, extensibility and privacy.

Sensing - If an environment is to become intelligent (whereby its responses are similar to that of a human) then it needs to respond to users' actions and voice commands, gathered from various environmental data sensors. This sensing occurs on a wide scale; from simple IR sensors, which can be used to turn on lights when somebody enters the room, to video cameras, which can be used to measure a whole host of variables. Audio-visual recording techniques are especially important because of the large amount of information that can be extracted from the data stream. For example, in Michael Coen's Intelligent Room at MIT's AI Lab [125] video cameras have been used to track people and in Lucente's Visualisation Space at IBM Research [126] they have been used to understand gestures (as discussed in section 2.4.5).

User Interfaces - The EasyLiving environment uses a mix of traditional desktop PCs and more advanced user interfaces such as touch screens. In general a user will choose an interface mode constrained only by the devices available. However, they are not limited to a single device. This migration between different user interfaces empowers them to move between different rooms and activities while always using the most appropriate method of interaction.

Extensibility - Automatic extensibility has been built into the EasyLiving architecture and this enables the system to automatically incorporate new devices as they are added [147]. All the rooms within the EasyLiving environment have a dedicated *room server* and this holds information on the model of the room, including its geometry, its contents, and locations of people. All room servers are connected to a main central server, from which they download the software and information that is global to the whole system, such as the current time and a directory of people.

Privacy - *"In any intelligent environment, there is a trade-off between privacy and convenience"* [147]. Having a room full of cameras and microphones raises important questions such as who can watch and listen to experiments and who can access experiment recordings. The collecting of more abstract data such as user habits and preferences could, over a sustained time period, enable someone to build an incredibly accurate profile of a subject. Additionally, the ability of the environment to collect all this data passively, while it is not in active use, raises further concerns. Does the person

who cleans the space at the end of the day give their permission to have the daily routine recorded and potentially scrutinised. In the end there is never one single answer to the question of privacy. The EasyLiving environment encrypts all data and this is a good idea, but in the end having an open system, which can record all the required data, while at the same time convincing subjects their privacy is secure is a very difficult thing to achieve.

EasyLiving is an excellent example of physical simulation environment that has been built to fuse together heterogeneous elements without the need to build an entire smart home. The project was purely research based and this allowed the environment to take a more abstract role. Traditional everyday objects, which you might expect to find in the average home, were not always present in the EasyLiving environment (e.g. a bookshelf full of books). The project does, however, demonstrate the ability of a simulated environment to provide a valid platform for developing prototype architectures and technologies. The EasyLiving project does have limitations that hinder it from being used to investigate smart home control interfaces. The most prominent is the focus of the project on intelligence. The space has been built to learn and respond to users in an autonomous fashion and it therefore does not facilitate user-based control. This type of automation is not required for user experience based research as other alternative techniques are available (e.g. Wizard of Oz). The autonomy also leads to challenging design and configuration issues increasing the cost and complexity of the space.

2.6.5 Cooltown

The Cooltown project offered *"a Web model for supporting nomadic users, based on the convergence of Web technology, wireless networks and portable devices"*. The project was built entirely from web technologies and can be seen as a forerunner to the *Internet of Things* paradigm [46] [48] (discussed in section 2.1.1). Instead of focusing solely on devices (e.g. IoT centres around everyday objects), Cooltown focused on the nomadic user, specifically how a layer of infrastructure can support a user's mobile devices. It was developed by HP Labs and culminated into a set of physical ubiquitous environments that investigate the users experience and the integration with web-based technology.

Traditionally a user would sit at a desktop computer, wired into larger network, interacting with the computer through a single point. Today's user can be nomadic, moving from one place to another, while all the time interacting with some form of computing device. Currently this would be via a smartphone or laptop computer, which utilises the larger wireless networks and internet – ubiquitous in modern society. Cooltown looked at how to dynamically create a web representation for every person, place or thing. The developed architecture *"enabled the dynamic generation of web contents based on the user context (location, identity, device capabilities), on his security permission and on the relationships with other web presences"* [148]. Using this web presence a user can access a wealth of previously inaccessible information, for example, imagine a patron enters a museum carrying a smartphone. The museum has web pages available for each room; upon entering a room the corresponding webpage automatically becomes available to the patron's smartphone. By approaching a painting the corresponding information also becomes available along with a service to send a postcard to any address in the world. The postcard will depict the painting and short message that is entered by the patron while they are still looking at the painting for inspiration. The postcard can then be sent and a micropayment could, theoretically, debited from the users credit card.

Cooltown foresaw this short scenario as the future, and the current movement towards the IoT evidence for their accurate prediction. As a method for researching types of control interfaces for smart homes, the project has many limitations. The Cooltown project was built entirely from web technologies and was focused upon how these technologies could technically enable a convergence into the *Internet of Things*. There was limited focus on how these technologies would deliver not just an enabling experience but how that experience could be user driven in an intuitive, easy-to-use manner. New technologies in particular HTML5, CSS and JavaScript and the advent of native mobile apps could be a useful addition to this research, as these types of technologies could bridge the gap between the enabling technology and user control interface. Examples of the physical ubiquitous environments implemented by the Cooltown project are also at odds with the intimate relationship users form between themselves and their home. The Exploratorium museum was the setting for a Cooltown

experiment [149] and it was here that, "*visitors carrying wirelessly connected devices were given opportunities for exploration, sharing, explanations, context, background, analytical tools, and suggestions for related experiences"* [149]. The Cooltown work did not focus on the usability of devices or interfaces, but rather on the layer of infrastructure required to support the nomadic user with wireless handheld devices. The location of the experiment, the Exploratorium museum, is also fundamentally different from the average home. The building is very large with many open spaces and exhibitions. This produces different interactions between a user and the space compared to a domestic environment. Although Cooltown used control interfaces, usability was not a high priority of the project. Capturing any data relating to how an interface was used was very difficult. No data logging or empirical capture techniques have been included within the research. On a higher level the project does demonstrate the experiences a user may have while interacting with a ubiquitous environment.

EasyLiving and Cooltown demonstrate the usefulness of a physical simulation environment. The main limitations of these types of environments are the relatively large development time, cost and lack of focus on user interfaces and control. When iterating interface prototypes, time and cost are two important constraints that need to be kept to a minimum. A technique for overcoming these shortfalls was identified by the EasyLiving research team, the Wizard of Oz technique, but there is no evidence to show that it was ever implemented. Even if such evidence existed, the focus of the space on intelligence is at odds with the idea personal control through the use of a device or interface (as outlined by Koskela et al. and discussed in section 2.4.5). EasyLiving is, however, a useful project to use as reference for a Wizard of Oz based simulation environment. Such an environment would instead focus smart home control and evaluation. It would be similar to EasyLiving, in that it would centre around the use of a physical space where users could be observed, but the focus would be on the use of the Wizard of Oz technique to provide a mechanism for users to control simulations with tangible objects, e.g. smartphone.

2.6.6 Video Simulation

If the use of simulations within a generic physical space can be used instead of a physical functioning smart home, the type of simulation required to convey a functioning device, needs to be determined.

The use of the Wizard of Oz technique will provide a means of controlling both an environment and simulated devices. Instead of being physical objects these simulations would be pseudo-realistic representations of devices commonly found in a domestic environment. Within the current literature the visualisation of physical objects has been achieved using various means. They could be visualised using a real-time rendered 3D model, a video, a photograph, it may even be a sketch. It could be displayed on a CAVE [150], Head Mounted Display (HMD) [151] or wall monitor. What level of immersion or visualisation is required? A study by Bowman et al. [152] found that for less complex situations there was no real benefits to using a high-level of immersion. For although high-levels of immersion do have benefits (such as greater spatial understanding), they argue that this may be redundant for all but a limited number of cases. The use of a highly immersive system is therefore not always necessary and there can be many benefits gained from avoiding such costly or wasteful situations [152]. Conversely Sutcliffe et al. [153] found that CAVE environments were remembered better, had better usability and provide a better sense of presence to users. A study by [154] Johansson and Ynnerman [154] found no significant difference from the use of a desktop system and a fully immersive environment. Kasik et al. [155] also supported this finding when they asked users to perform a 3D navigational task. Here, a 20" monitor was the device that produced the best performance; however, there was generally little difference between display technologies in regards to performance. Swindells at el. [156], who also found little difference in performance between display devices, noted *"display type and rendering style were less important than factors such as task structure, navigational context, and individual differences"* [156]. There appears to be no universally accepted answer on the level of fidelity required to increase users performance of a task, or to provide them with an experience considered *immersive*. A decrease in fidelity and the use of monitors does generally seem to have little difference

in performance. Furthermore, in a limited number of cases they have even been found superior for inducing user performance.

Mania et al. [157] go as far to state that a lower fidelity environment, specifically the quality of the rendered graphics, may even enhance the memorial experiences associated with an object due to the additional demand placed on cognitive systems "*because of its novelty or variation from 'real'*" [157]. It seems therefore that the use of low quality environments and graphics has little detrimental effect on the outcome of performance from users, and that focus should be spent on task structure rather than the development of a photo realistic virtual environment. Furthermore, the use of *lower fidelity* immersion, e.g. the use of the monitors rather than CAVEs may even allow for the development of a more interesting experience. Monitors are, after all, still devices that can be physically moved, turned and covered over in a similar way to many of the physical devices that they will be used simulate. The use of lower fidelity graphics will ultimately also reduce the amount of time and effort required to develop such simulations.

The use of monitors and video will provide a means of visualising simulations. Video enables a wide array of visualisation techniques and formats to be used within the environment. It is a medium that can simultaneously display 3D environments, photographs, sketches and real-world recordings. Although video can display various levels of fidelity, it is envisaged that the use of lower fidelity simulations will provide a high-level of user engagement while at the same time promoting rapid development and configurability of simulations. Video can also be updated and controlled in real-time from a remote location enabling simulations to be modified by a wizard based on user interaction.

2.6.7 Summary

In summary, simulation is a useful tool for collecting information that can benefit the production of an idea, system or object. That idea, system or object does not have to be the end product; rather it can be a higher-level process, in which, other products are integrated thereby enabling the results from one simulation to be used within another.

As its end goal is a reduction in time, cost or effort and an increase in understanding, quality and efficiency.

Within the context of this thesis, simulation will be used as an alternative to building a physical smart home. As outlined in section 2.6.2, virtual simulation will not offer the physical interaction required between the user and the control device. However, basing the environment on a physical simulation and on the work performed by the EasyLiving project would provide the appropriate foundation for the development of an environment focused on smart home control. The implementation of the Wizard of Oz technique and the use video are highly appropriate for providing a platform that enables simulated devices to be developed and deployed within such an environment. They will also facilitate the real-time modification and control of such simulations. For the subject this would create a similar experience to interacting with a smart device in a smart environment. Replacing the automated system intelligence with a human operator would also reduce the overall effort and cost required to develop such as simulations.

2.7 Overview of research problem

The literature discussed in this chapter has outlined the four problem areas preventing the mass adoption of smart homes and smart technologies as:

- Cost
- Interoperability
- Retrofitting
- Usability

Of these, usability is considered to be a key problem area [10, 28, 29] and forms the scope of this thesis.

In order to understand the type of control interfaces that need to be improved, a detailed analysis has been undertaken (discussed in section 2.4), which reviews research related to smart home control. From this, it is reasonable to argue that smartphones are the preferred control interfaces, and the transfer of poor user experience from previous control interfaces to this new device is a prominent issue.

When considered within the broader smart home context, the current literature lacks the type of information that could be used to improve the current generation of mobile touch screen interfaces. Furthermore, the tools available to researchers for gaining an insight into how the usability of a smartphone interface can be improved are limited in both their functionality and output fidelity. As discussed in this chapter, smart home usability is a major concern for users [10, 16, 24]. With the preferred smart home control device being a smartphone, how the interface of such a device can be improved, in the context of smart homes, is an interesting area for study.

What can be gained from the presented literature is a method for improving user interfaces and user experiences in general. The commonly accepted workflow used to define, design and implement the user interface (or user experience) for computer-based user-driven applications is one centred around the end-user. Within the workflows prototyping is presented as a valid solution for improving user interfaces or more specifically, the iterative process of testing a prototype with a user, then refining the interface based on the data gathered, facilitates an interface improvement [99].

The research presented in the literature review also outlines the need for a physical environment to evaluate prototypes for smart home control. Although a functioning smart home would allow for the most accurate results it is extremely expensive and unnecessary for all but large projects where the exact environment is paramount to the success of the research being conducted. Simulation is presented as a valid alternative to building a physical smart home for conducting smart home usability research. This is not to say that a physical environment is not needed, but rather, that this environment could be a physical simulation of a smart home, specially designed to facilitate the evaluation of the iterative prototyping process.

To give a better understanding of the research problem the following conclusion is presented:

The stimulation of smart home adoption requires an improvement of iterative prototyping techniques for smart home control interfaces. A valid interface to focus upon is a smartphone interface. Any improvements based on user centered iterative prototyping techniques should involve evaluations with end-users located within a

physical environment. This physical environment may be simulated but there is always a need to have real people in a physical space operating tangible devices. The subject may be interacting with what they perceive are real smart home devices but which may be a pseudo-realistic simulations of the real world.

This thesis will attempt to improve prototyping and evaluation techniques for smart homes in line with the above conclusion. The approach taken will not focus on the creation of generic smart home user interface components or the absolute resolution by users regarding their wants and desires; as this is a broader issue that is out of the scope of this thesis. Rather, it will focus upon the creation of prototyping and evaluation tools and technologies that will empower other researchers and users to investigate the issues.

With this type of approach a complete solution to the broader issues described above can be facilitated. The alternative, disregarded approach, would produce useful results, but these would be on a micro-level, would not be culture specific [36], and would become obsolete very quickly.

This thesis will therefore focus on the development of a framework, which includes a number of novel components aimed at facilitating prototyping and evaluation techniques for smart home control. This framework will be evaluated as a round-tripping toolchain for prototyping smart home control interfaces. It is out of the scope of this thesis to gauge the lasting effects of the framework on the actual adoption of smart home technology, as this could take many years. However, with the improvement of prototyping and evaluation processes, researchers and designers will be empowered to make smart home UI improvements. Ultimately it will be this subset of people that will make the conclusive changes.

Chapter 3

Prototyping and Evaluating Mobile Interfaces for Domestic Environmental Control

This chapter is divided into two main sections. The first introduces the concept of a framework for the design, prototyping and evaluation of smart home control interfaces. This includes three main framework components, the Reconfigurable Multimedia Environment (RME), Simulated Interactive Devices (SID) and iProto. With a reflection on the current literature, the second section discusses the aims of the framework allowing for the creation of a set of requirements.

3.1 A Framework for Prototyping and Evaluating Smart Home Control Interfaces

Section 2.7 gave an overview of the research problem and presented the focus of this thesis; the development of a framework for prototyping and evaluating smart home control interfaces.

The framework will be split into three major components; the development of a novel prototyping tool for smart home control interfaces, the development of a novel space for conducting user centred research relating to smart homes, and the development of a novel tool for simulating devices normally found in a domestic environment.

Figure 3-1 [a] shows the framework being used to produce a prototype interface for a smartphone device. Figure 3-1 [b] shows the prototype interface being evaluated in the user centred evaluation space aided by a simulated interactive device.

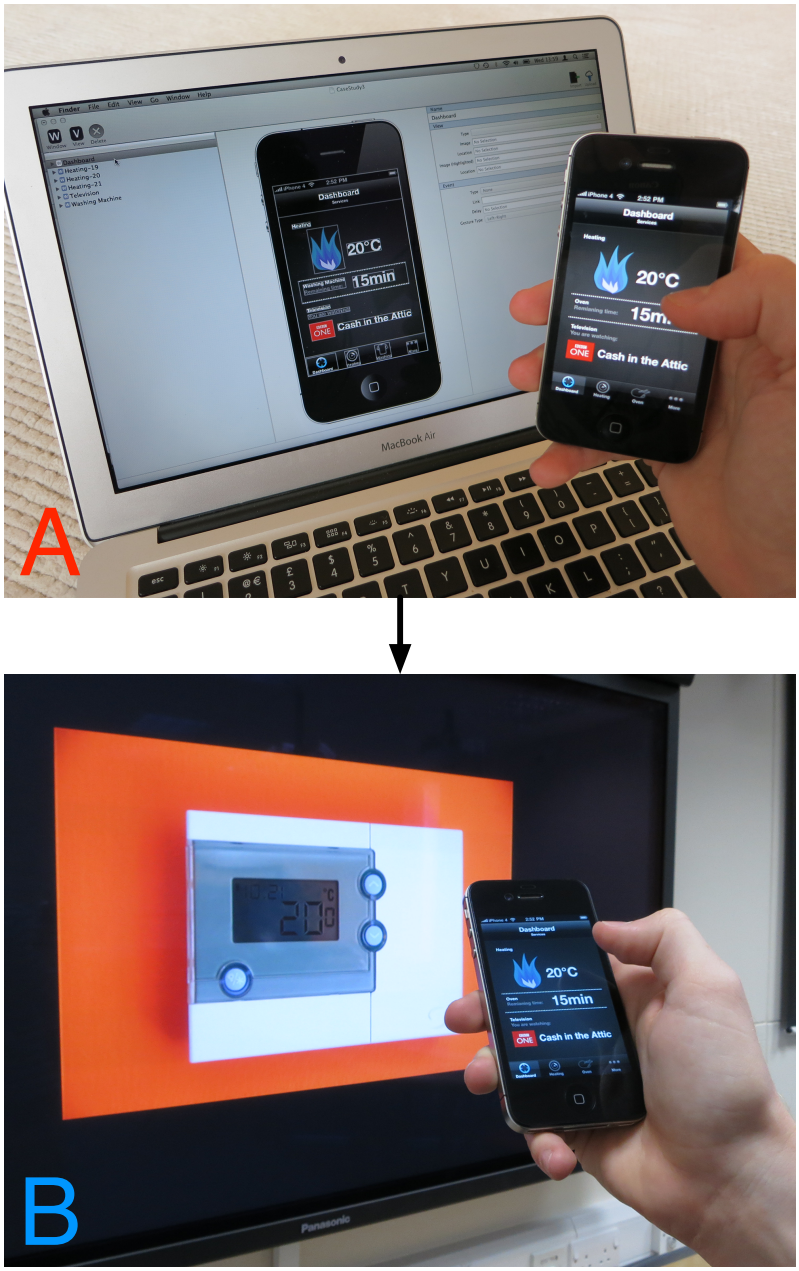

Figure 3-1 - [a] Prototype Authoring and [b] Prototype in use with SID being captured with RME

The first of the three elements, the novel prototyping tool, will be used to facilitate the improvements related to smart home user interfaces and is introduced in this thesis as iProto. The second element, the novel reconfigurable research space, will be used to perform the iterative user centred evaluation prototyping process and is introduced as the Reconfigurable Multimedia Environment (RME). The final element, the device simulation tool, is used to produce simulated representations of physical interactive devices found within the domestic environment and it is introduced as the Simulated Interactive Devices (SID) tool.

Figure 3-2 shows a high-level overview of the framework. Walking through Figure 3-2 from the top right, a prototype is first produced and authored using a graphics application and the iProto toolchain. This is then deployed onto an end device via the cloud distribution mechanism. The user then interacts with a prototype UI (Figure 3-2, centre left), which they use to control SIDs (Figure 3-2, centre). These SIDs are not really controlled from the prototype; rather, they are controlled by a wizard using a new parallel control paradigm (Figure 3-2, centre right). Finally all the interaction between the subject, the smartphone and the SIDs is captured using an audio/visual capture system (Figure 3-2, bottom)

Using this framework, a prototype can be produced and deployed onto a physical smartphone. This prototype can be used by a subject to control simulated devices, facilitated by the Wizard of Oz technique, in a dedicated reconfigurable evaluation environment. The resulting interaction is also captured for real-time or post analysis.

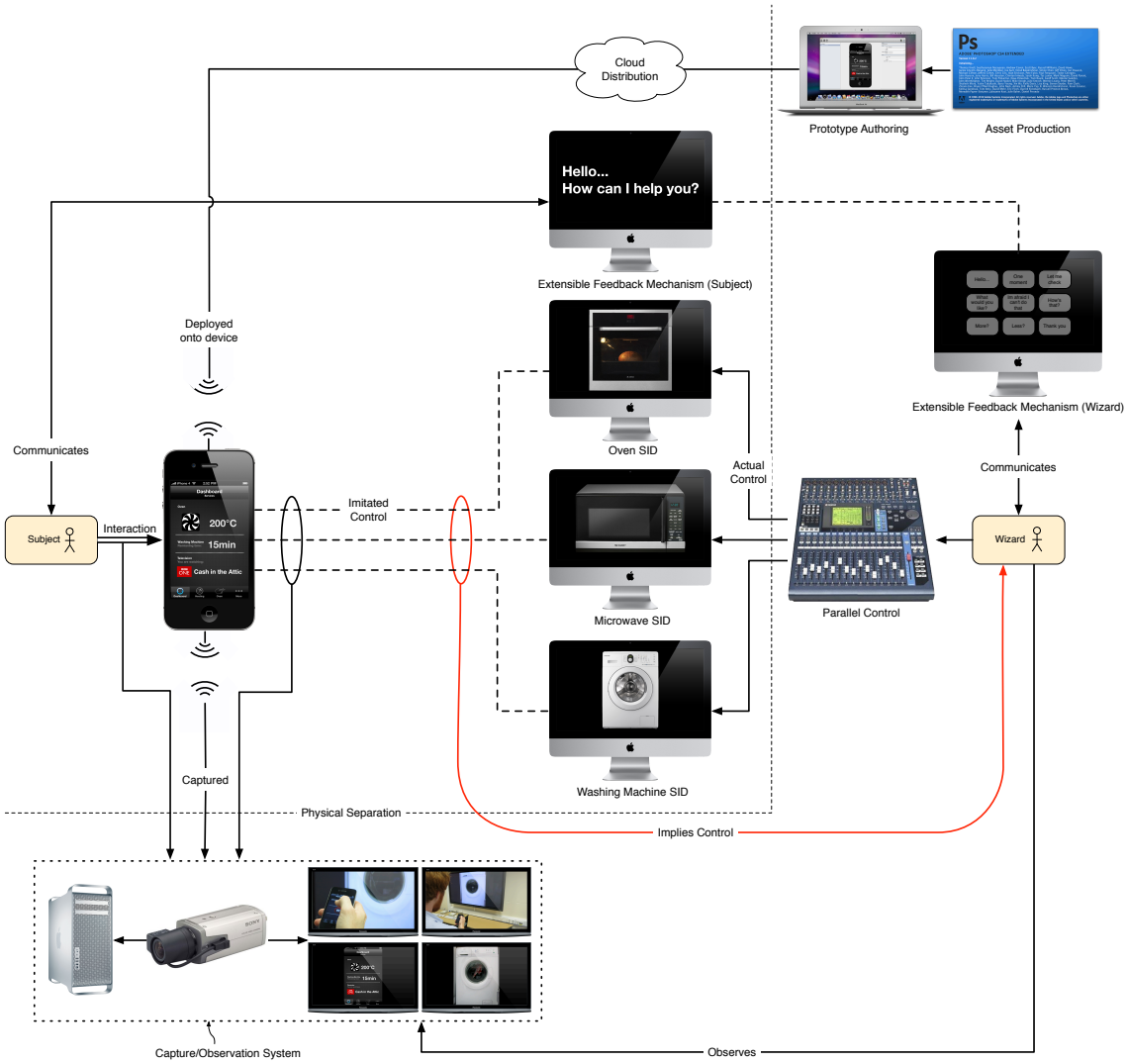

Figure 3-2 - High-level Overview of Framework

An example scenario is given below to provide a greater understanding of the framework.

John would like to develop a mobile application to control a number of devices located in a kitchen. John is not concerned with the technical aspects of this control, e.g. communication protocols, as these are specified by the manufacturer. John is really concerned with how to make the application as intuitive as possible for the user. He wants to test various interfaces and interaction techniques without having to first

implement all backend technical aspects. John would also like to capture the interaction between the user and the device to aid his designs.

This scenario would be modelled and simulated using iProto, the RME and a number of SIDs. Using the framework, John can design and produce a prototype interface and deploy it onto a real smartphone. He can also design and produce simulated devices for the subject to control with the prototype, via a wizard. Finally, any overt interaction between the subject, the prototype and the simulated devices can be captured through audio, video and logging.

Although the RME and SID can be used to simulate the scenario above, the type of logic used within a functioning smart home to respond to user interaction is missing. For example, the subject can use a prototype to increase the temperature of the oven but no signals are sent. For the model to be valid, the environment must facilitate interaction and feedback and it is envisioned that the Wizard of Oz (WOz) technique (discussed previously in section 2.6.3) would be the most appropriate solution.

In its simplest form, a WOz experiment involves replacing a computer system, partially or completely, with a human [140]. The same idea can be applied to simulation experiments "*whereby interaction is mediated by a human operator, the wizard, with the consequence that the subject can be given more freedom of expression, or be constrained in more systematic ways"[140]*.

Within the RME the WOz technique is utilised extensively to convince subjects into thinking they are controlling the simulated devices. Its use enables rapid design, prototyping and evaluation of smart home user interfaces.

The RME is a physical space where interfaces can be evaluated with the capture of quantitative and qualitative data. The environment allows for the simulation of smart home devices (with the use of the SID tool) enabling interaction between the subject and devices to take place during evaluation sessions.

iProto is a novel toolchain for creation, deployment and round-tripping of prototypes for smartphone devices. It facilitates rapid and iterative development of interfaces by interface designers.

The RME and SID tool can be used in conjunction with iProto thereby creating a complete framework for the design, prototyping and evaluation of smart home interfaces for domestic environments. As shown in Figure 3-3, iProto will facilitate the design process while the RME and SID will facilitate the evaluation process. Results from evaluations within the RME can be fed back into the interface designs produced by iProto thereby enabling the round-tripping of interface designs for smart home control.

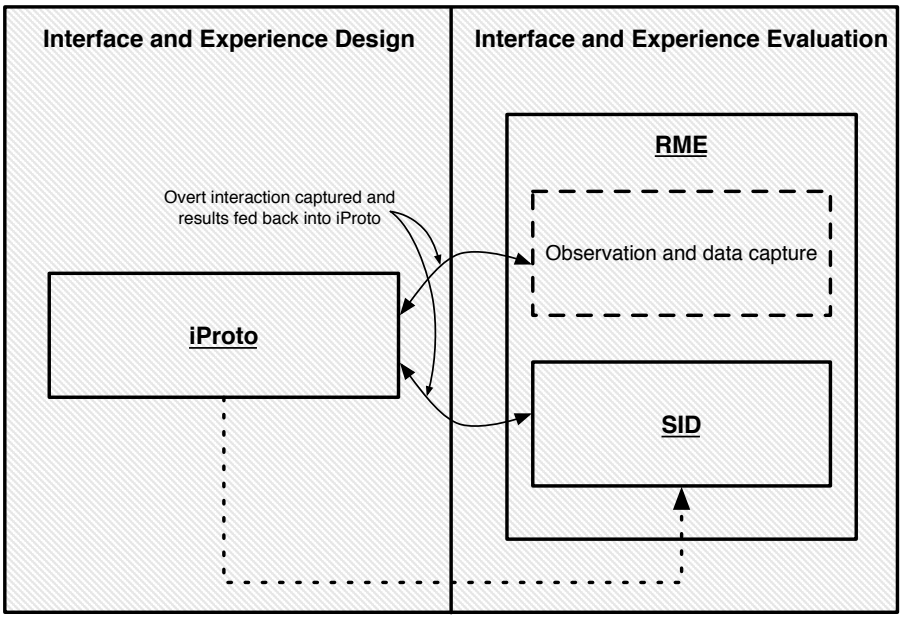

Figure 3-3 - Creation of a Framework for the Design, Prototyping and Evaluation of Mobile Interfaces for Domestic Environments

Whilst the components are designed to be used within the complete framework, it is possible to use them independently. In this instance the RME could be used to evaluate user needs within the context of smart homes, but without the smartphone interface. An example would be the use of NLP research or gesture control whereby the subject would ask the environment for a change (e.g. increase oven temperature) and the simulated device would respond accordingly (with the use of the WOz technique).

3.1.1 A Novel Design Methodology For Mobile App Prototyping

The current tools available for the production of smartphone prototypes, facilitate the generation of low fidelity, non-gesture based interfaces with minimal support for interaction. They do this through the use of separate graphics applications and the production of assets that are abstracted away from the main production process (discussed in section 2.5) following the design cycle shown in Figure 3-4 [a].

The basic layout of any application interface will usually be defined using wireframes and concept art [96]. Using these, prototype designers will produce the prototypes, including the prototype graphics, which may enter an evaluation process. Using this, information designers will then generate a new set of graphics, including general assets and mock-ups. These will then enter another iterative process with the clients and more users. The process promotes the generation of low-fi prototypes that are visibly unlike the final interface, and although these are useful, they do not provide the level of fidelity required for the user to make the conceptual leap between the prototype and final design [116].

It can therefore be argued that the current design cycle model twinned with the limitations of the currently available tools (outlined in section 2.5.6) hinders the speed and quality of the prototypes produced for smart home control. A new methodology is therefore proposed that enables the use of high fidelity prototype assets and touch screen interaction (shown in Figure 3-4 [b]). Furthermore, the methodology will facilitate the iterative prototype process allowing for subtle changes to be made to graphical assets that can then either be fed back into the prototype cycle or incorporated directly into the final production interface.

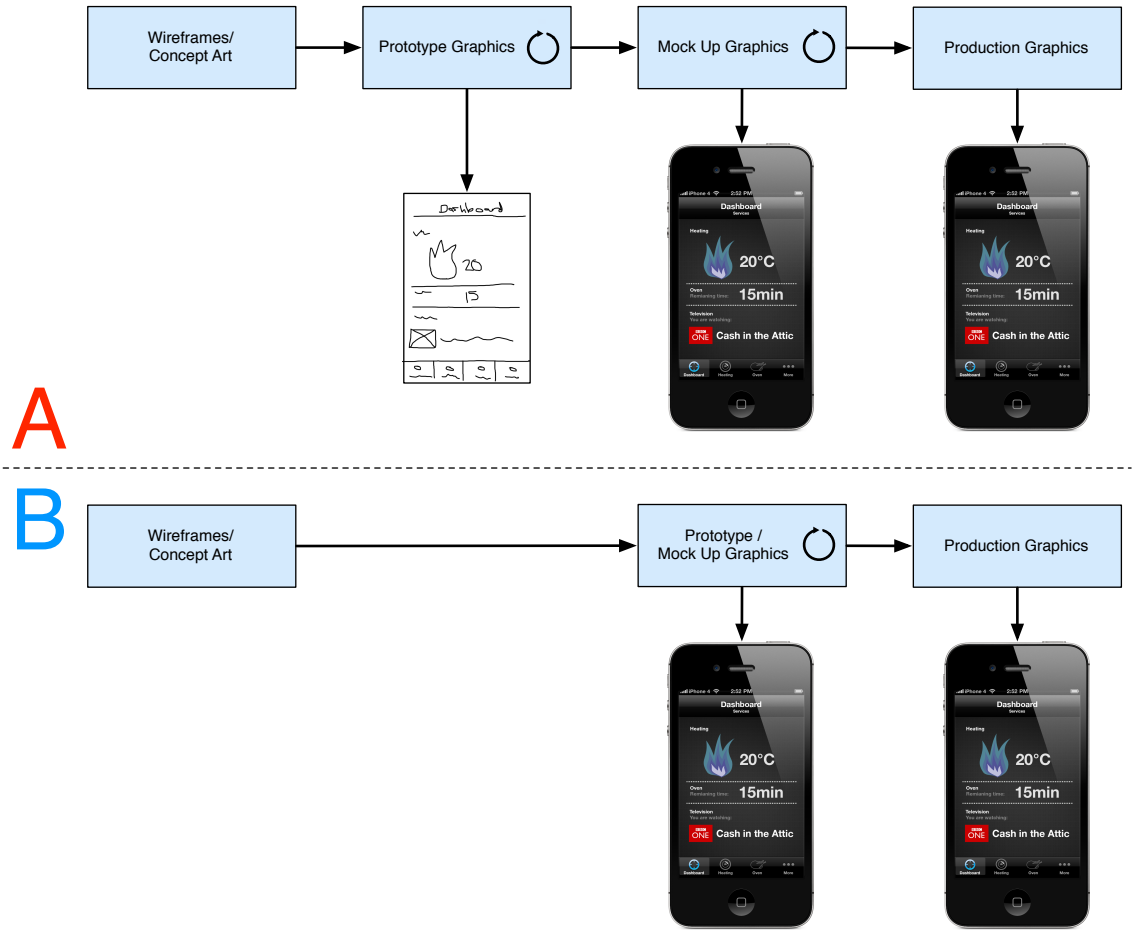

Figure 3-4 - [a] Current Mobile Application Design Cycle Model and [b] Modified Mobile Application Design Cycle Model

3.2 iProto

Figure 3-5 shows a very high-level overview of the new design cycle model. The proposed cycle reduces down into the three main stages: first, the creation of assets (either after or with the creation of wireframes/concept art). Second, the utilisation of the graphics from the first stage in a prototype construction tool. Third, the wireless deployment of the prototype to an end mobile device. The key idea is the ability to use the same graphics for both the prototyping and production interfaces while also facilitating rapid iterative user centred evaluation. Once the prototyping evaluation has been completed the same graphics can be exported and handed over to the development team for incorporation into the final production interface.

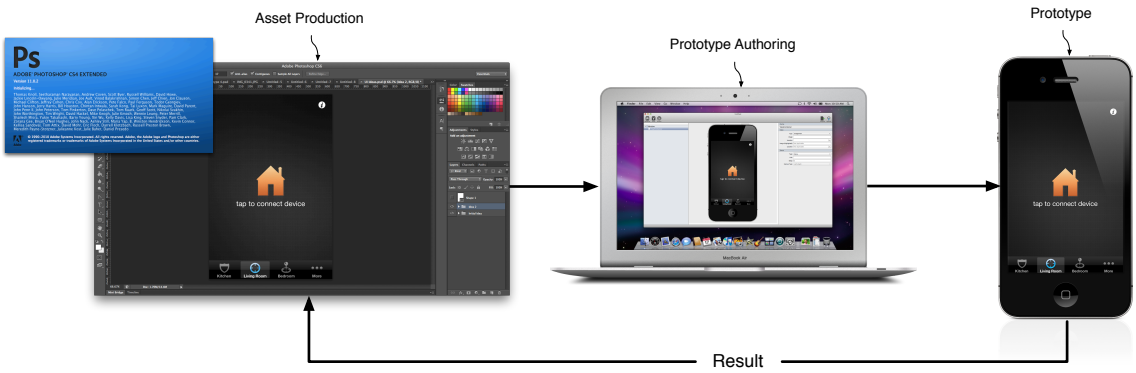

Figure 3-5 - High-level Overview of Proposed Design Cycle Model

The simplification of this design methodology over the current process, along with the development of a new tool, enables the aforementioned limitations to be mitigated. It also brings consistency to the asset generation process; using the new model all the graphics (either prototype or production) can be produced in the same graphics application. This streamlines the process of asset generation into a single entity, reducing the time needed to produce not only assets but also the entire app. Additionally, results from the prototyping process can instantly be fed back into both production and prototype graphic generation.

Figure 3-6 shows how the proposed design cycle model would incorporate a distribution mechanism. In essence, the iProto toolchain would allow the prototype interface to be stored in the cloud where it could then be pulled by an unlimited number of devices located anywhere in the world. A number of different prototype interfaces could also be stored within the cloud providing a platform for quickly and easily testing radically different interfaces or for minor changes.

As shown in Figure 3-6 the results from prototype evaluations can be fed back into the designs of the actual prototype. However, for this to be accomplished the results first need to be captured and analysed.

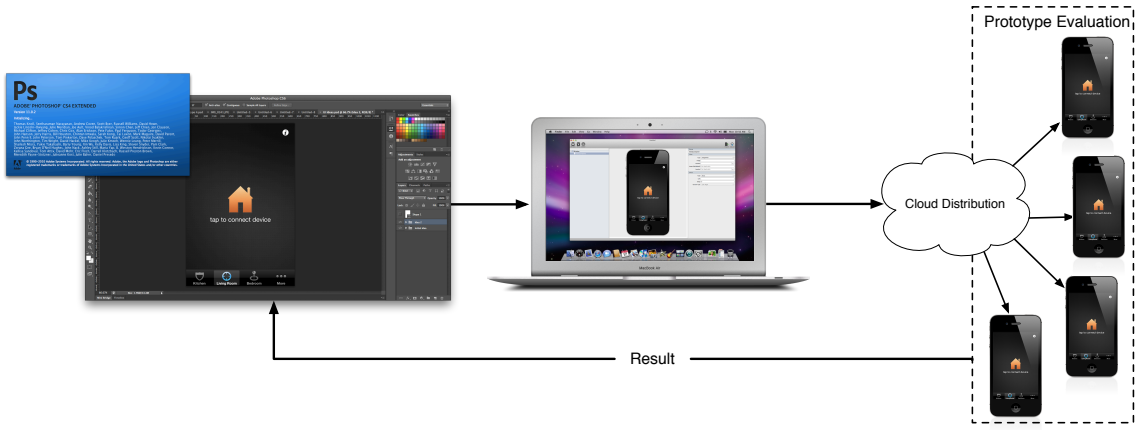

Figure 3-6 - High-level View of the Proposed Design Cycle Model with Cloud Distribution

3.3 Reconfigurable Multimedia Environment

The Reconfigurable Multimedia Environment (RME) is a space where a physical simulation model can be created, where devices can be evaluated and where quantitative and qualitative data can be captured in real-time for analysis. This model represents a smart home in which a user can physically manipulate their surroundings using a mobile interface. It is proposed that the system configurability allows for rapid changes to the simulation model, thereby allowing an increase validity of evaluations conducted.

The overall aim is the creation of a space, which will allow researchers to evaluate devices and interfaces to better understand the requirements of users in relation to controlling smart domestic environments. Facilitated by the iProto toolchain, this information can then be fed back into the interface designs. The initial aim of the RME is to provide a space that can simply capture interactions between a subject and prototype interface. It is envisioned (based on the current literature outlined in section 2.6) that video capture and logging would be best suited to this. This interaction will be analysed, either in real-time or post-evaluation and the resulting information fed back into the prototype designs.

Figure 3-7 shows a high-level overview of the RME. The overt interaction between the subject and the smartphone interface is captured through audio and video recordings,

along with the control between the smartphone and the simulated devices. In addition to this, the actual smartphone interface will also be captured via wireless transmission.

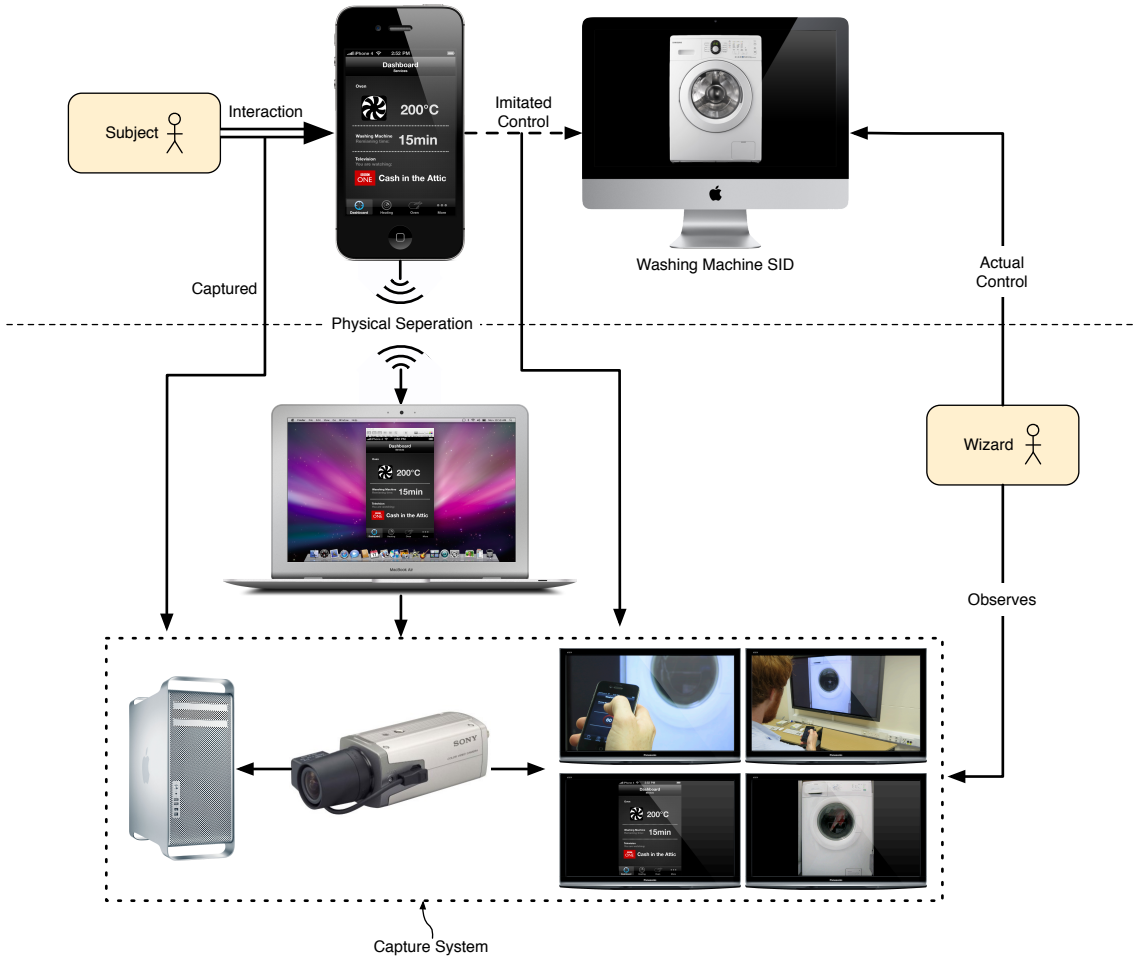

Figure 3-7 - High-level Overview of RME

Because the prototypes will be used on a mobile device, it is important to not physically constrain the location of the subject during evaluation sessions. Connecting the phone via physical wires limits the subject's movement, therefore any solution designed to capture the actual smartphone interface should be wireless. The smartphone has been designed to be mobile and it should remain that way during evaluations. Capturing the interaction between the subject, the device and the immediate surroundings can be achieved using audio-visual recording techniques, which cover a wide physical area.

This solution is non-direct, as it relies on the use of cameras and microphones to capture a wide array of information, including the prototype interface and relevant interaction. With the use of such a solution capturing a mobile prototype interface is challenging. The device is relatively small and easily obstructed from the view of distant cameras. Moreover, the difficulties in viewing the intricate details of an interface, along with the control movements (e.g. gestures) performed by the subject limits the use of a non-direct solution. The use of an additional direct capture solution is therefore proposed that would wirelessly integrate the RME with the mobile device enabling the actual smartphone screen to be captured and viewed by the wizard in real-time.

Figure 3-7 also shows the inclusion of a SID being controlled by a wizard. In section 3.1 an example scenario outlines the type of research that could be carried out using the framework. In the scenario a number of devices are required for interaction, for example, a washing machine, oven or light. These typical devices would be commonplace within a regular domestic environment. All therefore need to potentially be included within the RME so that during evaluation subjects can interact with them.

Obviously one possible solution would be to buy and keep the devices in storage until they are required. Aside from the cost issues this would incur, it is relatively impractical. Moving a large object like a washing machine would be very difficult, especially considering the need for the RME to be rapidly configurable. Moreover, as new devices are released there would be a need to constantly replace or upgrade existing ones, incurring more cost and effort in the process. Due to these issues a new approach is therefore required.

Within the RME, the reason for including devices is to enable the generation of requirements for smart home control interfaces. Based on the literature discussed in section 2.6.2 it can be argued that simulation would offer a valid alternative to the use of physical devices. The RME will therefore provide a platform for the simulation of devices found within a smart home. Depending on the scenario, different device simulations will be called upon and used with the RME. The tool that will enable the creation and control of these simulated devices is introduced in this thesis as the Simulated Interactive Devices (SID) tool.

3.4 Simulated Interactive Device (SID)

A Simulated Interactive Device (SID) is a video-based simulation of a physical device or control interface, normally found within a smart home. The simulation is presented to a subject on an image display, such as a computer or television, within a physical environment such as the RME.

For a simulation to accurately represent a real device found in a smart home, interaction between the subject and device must be supported. For example, if an interface being evaluated requires the user to switch on the television, the simulated environment must be capable of responding to that action in an appropriate manner. Furthermore, the simulation tool must be capable of responding to the needs of the subject, but also to the designer.

Within the framework the SID tool aims to allow the simulation of domestic devices for integration within the RME. These will facilitate interaction using the WOz technique, based on the literature review outlined in section 2.6.3. Having ruled out the use of physical devices, it is proposed that the SID tool be based on a digital representation of a physical object to utilise the benefits of digital technology. For example, a limitless library of simulated devices can be stored and used as required within a scenario. Furthermore, new simulations can be produced without the associated cost of purchasing a device, and unreleased or conceptual devices can also be utilised.

It is envisioned (based on research outlined in 2.6.6) that video would be the most appropriate medium for the visualisation of the simulations. With the use of video, any device can be mocked-up as either a static or dynamic simulation, or as a combination of the two. For example, an oven could be simulated with the use of a photograph of a real oven (a static element), and the controls (e.g. temperature setting) could layered on top with controlled video (a dynamic element). This video could then be manipulated in real-time depending on the interactions of the subject.

The use of multiple static elements could be used to create the effect of a dynamic simulation. For example, a desk lamp is a relatively simple object to capture and simulate. It could be photographed in an off state and an on state (static elements) then

composited one on top of the other. Using the opacity attribute of one image, the brightness of the lamp can then be variably controlled, as shown in Figure 3-8, thereby creating the effect of a dynamic simulation.

Figure 3-8 - Lamp Video Simulation [left-right] - On State, Various Composited Dimmed States, Off State

The potential use of multiple simulations also furthers the abilities of the RME and increases the validity of the evaluation exercises. Using many simulations a subject will have the opportunity to interact with a range of devices. Figure 3-9 shows a high-level overview of multiple simulations. Although the use of SIDs addresses the issues surrounding how a subject will interact with smart devices, it introduces problems with control. SIDs are based on the WOz technique and the use of a human mediator. If each SID requires the manipulation of a number of variables, all in real-time, and multiple SIDs are being used, a human operator could easily become overwhelmed. This would create an unrealistic experience reducing the validity of the simulations. The requirement of a parallel control mechanism is therefore desirable. This will enable a single wizard to control multiple variables across a number of SIDs. With the use of a wizard and a parallel control mechanism, many SIDs can be controlled in real-time. The digital representations of the physical objects will be generated with the use of desktop computers that will be located directly in the RME, or located in a separate area and configured to output the simulation to an external display located in the environment (e.g. a projector).

As can be seen in Figure 3-9, a wizard can use the parallel control mechanism to control a number of video-based SIDs being generated by numerous desktop computers. These may also be configured to output the simulation onto an additional display.

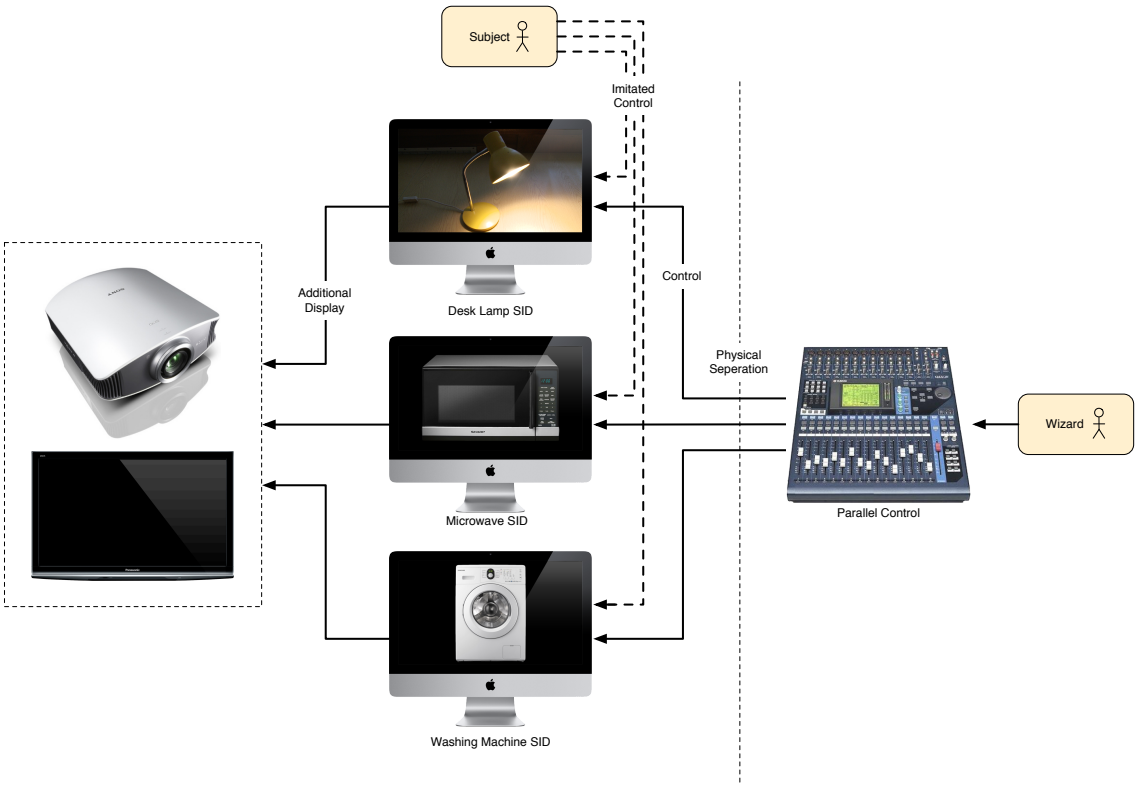

Figure 3-9 - High-level Overview SID

3.4.1 Extensible Feedback Mechanism (EFM)

In addition to simulated devices, a means of controlled two-way communication between the subject and a computer system or smart device would be a useful addition to the framework. This addition is based on requests, and feedback, from other research groups, e.g. NLP researchers, who are interested in using the RME for other studies. Specifically the request was for a text-based feedback mechanism that enabled two-way communication between the subject and computer system (or wizard).

The RME will therefore also provide a platform for the Extensible Feedback Mechanism (EFM). This will enable controlled two-way communication between a subject and a computer system or smart device. However, being based on the WOz technique the subject will actually be communicating with a wizard. The tool proposed

will therefore allow controlled two-way communication between the wizard and subject. This will be facilitated via configurable GUIs. The wizard is a human operator, but their role will be that of a mediator. Within a WOz study the subject must believe they there are interacting with a computer, not a human. The need for configurable GUIs is therefore due to the difference of abilities between a human and a computer. A wizard will may require a fundamentally different interface to the subject if they are to respond in an appropriate manner. For example, a wizard may require a set of pre-defined responses that can be immediately sent to the subject, whereas the subject might require a response box and keyboard display. Figure 3-10 shows a high-level overview of the Extensible Feedback Mechanism. It is proposed that the subject and wizard can communicate over a network with the use of separate desktop computers. This allows for the creation of different interfaces tailored to the needs of the individual.

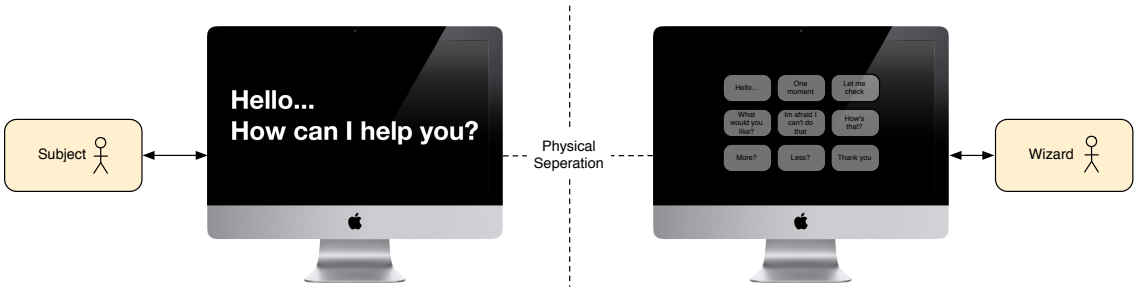

Figure 3-10 - High-level Overview of Extensible Feedback Mechanism

3.5 Framework Specification

It is at this point that all previous discussions are considered and summarised into a set of aims and specific requirements for a solution aimed at providing a design, prototyping and evaluation framework for smart home control interfaces.

3.5.1 Detailing the Aims

The specific aims of the framework will be detailed, setting the scope for the specification. By examining the needs of the user a better understanding of the functionality required of the framework will be gained.

The framework should be a tool for researchers. Specifically it should satisfy the needs of a designer, not the needs of a programmer. The purpose of this framework is to aid in the development of improved smart home control interfaces and, although these interfaces will at some point have to be programmed, it is not an issue to which the framework concerned. Instead the WOz technique will be used as extensively as possible to remove any programming requirements for the end-user.

The framework will support the use of the WOz technique via dedicated simulation and communication tools. The support will include a parallel control mechanism that will allow a single wizard to have control over multiple simulations in real-time.

Improvements to the preferred smart home control device, the smartphone, will be facilitated through a prototyping toolchain and iterative user centred evaluations. The prototyping toolchain will be sympathetic to the current design processes thereby further supporting the designer. It is important to state that this framework is not an attempt to define or stipulate *correct* user interfaces that should be adhered to by designers, but rather it is a tool to aid the creation of improved interfaces.

The observation and capture of interface evaluations will be supported through the development of a dedicated evaluation environment (the RME). This will be used by researchers to monitor and analyse information relating to user centred interface and interaction experiments. Results from these experiments will be fed back into prototypes thereby creating a complete framework for facilitating the improvement of smart home control interfaces.

A case study will be used to validate the framework through the round-tripping of a prototype for smart home control. It will include the design, evaluation and iterative prototyping of a smart home control app with the use of the RME, SID and iProto framework components.

3.5.2 The Requirements

The requirements for the framework are broken down into a set of statements, divided into three categories based of the three major framework components: the RME, SID and iProto.

1. RME

1.1. *Capture and observation of user centred evaluation experiments* – Running evaluation simulations will provide users with a wealth of quantitative and qualitative data that can be analysed. The framework should support the capture of this data in such a form that facilitates the use for either real-time or post analysis.

1.2. *Rapid configurability* – Having the ability to rapidly configure the space will allow users to quickly set up simulations for evaluations. This is especially important when considering the evaluations will centre around prototype interfaces that will be modified on a regular bases.

1.3. *Interaction and response* – Responding to a subject's physical interaction with a response is paramount to running an accurate simulation for evaluation. The response should be authentic in convincing the user that although they are interacting with a simulation, they are genuinely controlling part of the simulated environment.

1.4. *Feedback Mechanism* – The framework should support the ability for both the subject and the wizard to communicate with each other. This communication must be supported by different GUIs for both the subject and the wizard.

2. SID

2.1. *Multiple simulation models* – The framework should support multiple simulation models that can be deployed into a single environment. As interfaces or devices have a diverse range of

functional requirements, the simulation tool requires the ability to change and adapt to those requirements.

2.2. *Real-time control mechanism* – The framework should support a real-time control mechanism whereby a single wizard can potentially manipulate multiple simulations in parallel.

2.3. *Extensibility* – The framework should facilitate extensibility of simulations due to the need to incorporate, on a regular basis, simulations based on newly released devices and future un-released ones.

3. iProto

3.1. *Integrate into the existing application design model* – Any generation of assets for a prototype needs to complement current design models used by interface designers. Moreover, the production of a prototype should integrate with existing design workflows enabling a designer to make fine adjustments to an interface quickly and easily. This integration should also enable the production of high fidelity prototypes.

3.2. *Rapid design cycle* – The framework should facilitate a rapid design cycle for prototyping. This will enable the evaluation of either many different interfaces or a singular interface with many minor adjustments.

3.3. *Device deployment* – The framework should support the deployment of a prototype to an end device. Due to the intimate relationship between a user and a smartphone, only by evaluating the prototype on an end device with real users can full and valid data, regarding their experience, be obtained.

3.4. *Rapid wireless distribution* – To reduce the prototype iteration cycle time, a prototype interface will need to be quickly distributed to an

end device wirelessly to further reduce the time required for prototype deployment.

3.5. *Touch and gesture support* – The framework should support prototypes that can respond to touch (in the form of tapping) and gestures. All standard gestures need to be supported (a sample can be seen in section 2.5.5) as different tasks may require a variety of gesture input.

Chapter 4

A Reconfigurable Multimedia Environment

The Reconfigurable Multimedia Environment (RME) is a space for conducting user centred evaluations of smart home control interfaces with the capture of quantitative and qualitative data for real-time and post analysis. It also provides a platform for simulating a smart home environment using the SID and EFM tools. Figure 4-1 [a] shows the evaluation space, complete with cameras, screens, microphones and projectors. Figure 4-1 [b] shows the control room in which a wizard can observe interactions, route audio and video feeds, control SIDs and capture the relevant data.

Figure 4-1 - [a] RME Evaluation Space and [b] RME Control Room

Figure 4-2 shows an overview of the RME split into the two distinct physical locations outlined in Figure 4-1. The evaluation space is a physical environment where a subject can interact with a prototype and SIDs whist being captured by numerous cameras and microphones. A wireless screen sharing link between the smartphone and capture system will also allow the interface and interaction to be simultaneously captured directly from the phone. The control room is a separate physical space (hidden from the subject), from which a wizard can route the video and audio feeds, observe the experiments, define what is being captured for archive and control SIDs.

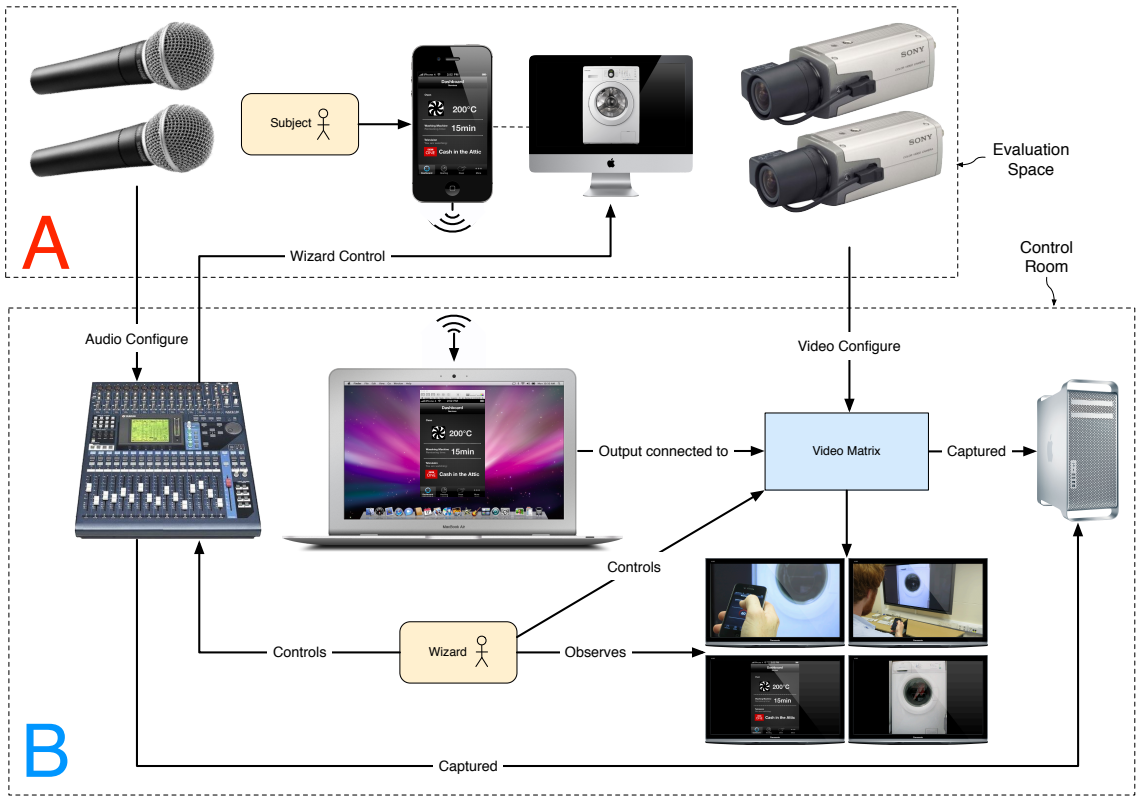

Figure 4-2 - Overview of RME - [a] Evaluation Space and [b] Control Room

4.1 Observation and Capture System

A specific aim of the RME is to observe and capture the interactions that take place within the space. The data captured should be both quantitative and qualitative giving the user a wealth of information to feed back into the iterative design and prototyping

stages. Due to the inconsistent nature of evaluation experiments the RME aims to capture large amounts of data. Depending on the specific session, parts of the data set will be irrelevant; however, having the ability to capture a broad array of data is paramount as it enables a wide variety of experiments to be conducted.

To satisfy the capture of quantitative data; information relating to any physical interaction happening within the RME will be logged and time stamped. Although not directly supported by the RME hardware, any software layer built on top of the RME will incorporate this type of data capture. Both the wizard and the subject will generate this data, for example when the subject taps on the button on a prototype interface the button name, description and time tapped will be recorded. Similarly, a *wizard's* response to a subject's interaction will also be recorded. The premise behind this type of data capture is similar to the ideas behind big data [158]. This is to capture and log all possible relevant data and allow it to be sorted and analysed post-evaluation.

The capture of qualitative data is more subjective. The prototype interface, or problem being evaluated will determine the qualitative method of capture. Methods of capture include interviews, focus groups, observations and audio and video recordings. Determining which method is most relevant is the responsibility of the user, usually based on knowledge of previous research within the area [159].

The problem therefore, is how to capture as much information as possible, both quickly and easily. A user can then determine what data is relevant for their particular scenario. Based on previous research (discussed in section 2.6.4), the main RME data capture will centre on the use of real-time video and audio. Using techniques usually deployed within a live television studio the aim is to provide users with comprehensive dataset of all actions happening within the RME. From this users can glean both qualitative and quantitative data for real-time and post analysis.

A physical space has been provided at the University of Sussex for the development of the RME. This space consists of a room approximately 8m wide, 6m long and 3m high. As with any video capture solution, the physical constraints of the space affect the possibilities of what can be captured.

Within the RME it is proposed that a multi-sensor network of microphones and video cameras is used to provide high quality data capture for user interaction analysis. Due to the availability of a finite number of cameras, the minimum number of cameras required to cover the space needs to be determined.

Figure 4-3 shows the physical space for the RME with a number of proposed camera locations. These camera locations have been selected based on the viewing range of each camera. If the camera is mounted on a tilting head it is possible to re-orientate its position, thereby increasing its viewing range and decreasing the overall number of cameras required. The pink circles shown in Figure 4-3 denote the range of movement possible by a specific camera. When this range of movement is twinned with its viewing angle it is possible to determine the parts of the space each individual camera will be capable of covering. The blue overlay shown in Figure 4-3 denotes the coverage of the space with the use of 9 cameras (the darker the blue, the more cameras are covering the particular area).

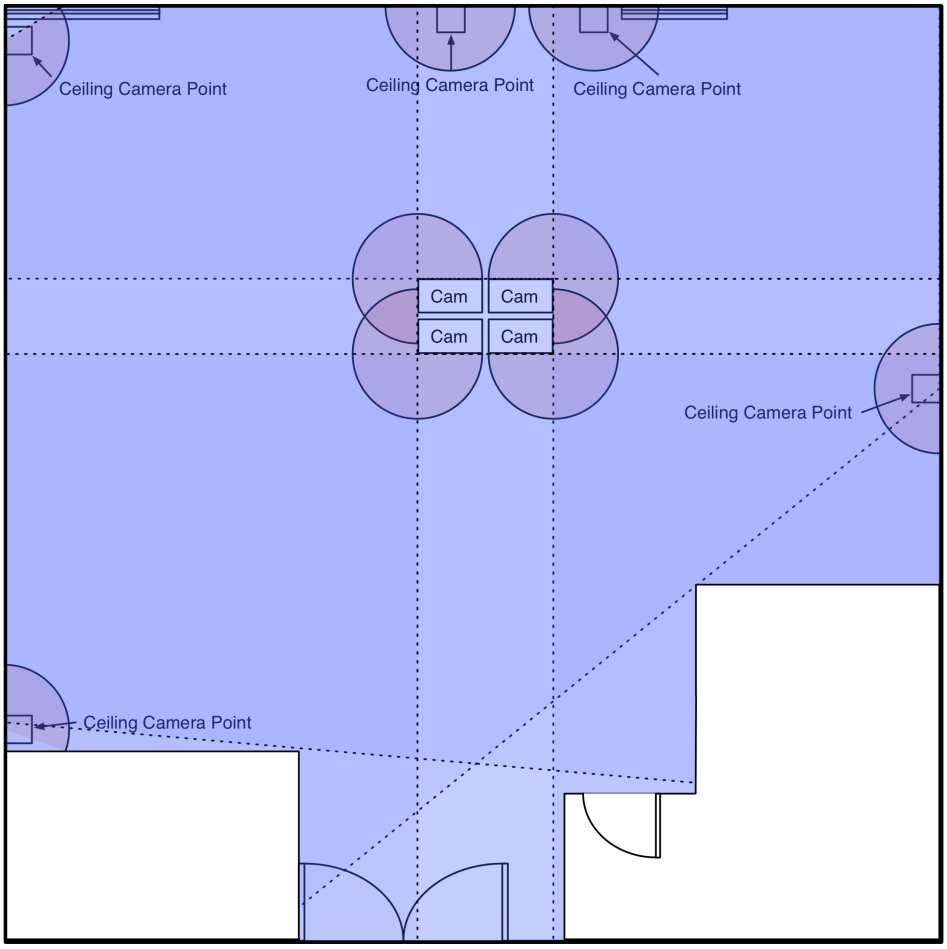

Figure 4-3 - Determining the Number of Cameras Required for the RME

It would be possible to cover the entire space with just a few cameras but this would not give the sufficient configurability and coverage. With the use of 9 fixed cameras the entire space is covered by a minimum of 3 different cameras thereby providing sufficient angles and feeds for satisfying both the configurability of the space and for capturing a wide array of interaction.

With the cameras in place, a means of viewing the feeds in real-time is required. This will enable users to observe multiple parts of the space from different angles, modify the scenario or analyse results during experiments. The proposed solution is a bank of monitors and a real-time routing mechanism. Using this, a user can change not just what cameras they are seeing but also what is being captured.

The premise of the RME is to create a type of real life fish tank. Within the confines of the space, users will be able to observe and capture subjects as they perform interactions. Although, in the context of this thesis, the space will be used to observe smartphone and smart home interaction, the space can be used to observe and evaluate any user centred research. This furthers the usefulness of the environment and enables it to be used for other types of smart home control research, e.g. gesture or voice control (discussed further in section 8.3). The RME will also provide a space for simulating devices found within a smart home. These will be produced using the SID tool.

4.2 The Physical Space

In order to observe and store a high quality data set, synchronised camera feeds are required. When observing and capturing from multiple cameras, delays between different feeds could introduce observational errors that will dramatically reduce the output quality from the RME research. For example, if a wizard is expected to respond instantaneously based on a subject's interaction and there is a delay in the observation feed, causing the response to be delayed, the knock on effects will have repercussions for the research results. To enable the synchronisation of all cameras within the RME, a dedicated 24v AC ring circuit will be installed. Having all the cameras synchronised to frequency of the same power supply allows for frame accurate synchronisation throughout the environment.

The need to capture multiple synchronised cameras is handled with a dedicated capture system capable of recording simultaneous feeds.

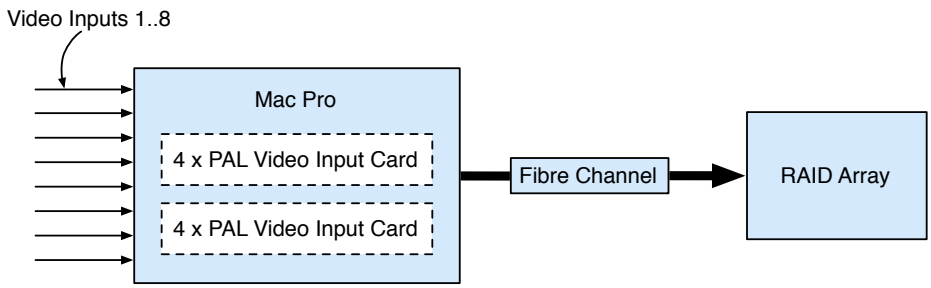

Figure 4-4 - RME Video Capture System

Figure 4-4 shows the proposed video capture system. The main workhorse of the system is a Mac Pro computer combined with 2 PAL video input video cards, each capable of capturing 4 simultaneous video feeds. To assess the capabilities of a range of hardware, in not only capturing, but also compressing multiple video feeds in real-time, the system was configured with different numbers of inputs at varying levels of compression. The final configuration presented in Figure 4-4 was found to be the best and most cost effective solution. This capture system design and configuration has ramifications for a number of key design decisions for the RME.

The first is the basic video format in use within the space. The main video formats used within the UK are Phase Alternating Line (PAL) and High Definition (HD). Although HD would provide a massive boost in quality, the capabilities of the top-of-range hardware in just capturing the video stream is severely limited. It was found that only 2 simultaneous HD video streams could be captured and compressed with a single Mac Pro computer. Multiple machines could be used; however, the cost implications to this are unsustainable. The reduced quality of PAL enables a more efficient throughput, allowing for a greater number of feeds to be captured and compressed simultaneously using a single Mac Pro. The use of a reduced quality video stream is the trade-off for an increase in the number of captured feeds. The final capture format and quality specifications can be seen in Table 1.

Format	Compressor	Resolution	Quality (Mbits/s - Variable)	Frames Per Second	Final Size (GB per hour)
Video	H.264	720x576	5	25	2.2GB
Audio	AAC	n/a	192	n/a	

Table 1 - Capture Format Specification

To increase the configurability and to further the extensibility of the space it is proposed that additional cameras can be connected and positioned on an ad-hoc basis. The

configuration of these additional cameras will not be defined and can be tailored according to the requirements of an experiment. A means of connecting equipment to the RME in an ad-hoc way is therefore required. The solution devised is the inclusion of a number of connection points located within the space. Using these points the user can select a location to connect the camera, then position it to give them the best results for their given scenario.

With the inclusion of these connection points the provision of audio sensors can also be determined on an ad-hoc basis. A general requirement of microphones is that they be near their sound source for the capture of high fidelity audio. By providing a means of connecting and positioning microphones in an ad-hoc manner, microphones can be moved and placed near the desired location. The aim is to give the user an element of control over the layout design of the sensors.

As with the cameras, there are also a finite number of microphones available. To allow for the best possible configurability, the RME will include a mixture of microphone types. Although the RME will be capable of observing many subjects, the finite number of microphones available limits the number of personal microphones for use. Based on the size of the space and the envisioned maximum number of subjects, 3 personal wireless microphones will be available. In addition to this, a number of omnidirectional table microphones, designed to capture audio from a larger group of people, and cardioid type, chairperson, microphones will be available should the space accommodate more than the envisioned number of subjects in a single scenario. Again, based on the size of the space and the maximum number of people that it can accommodate, a combined total of 7 non-personal microphones will be provided.

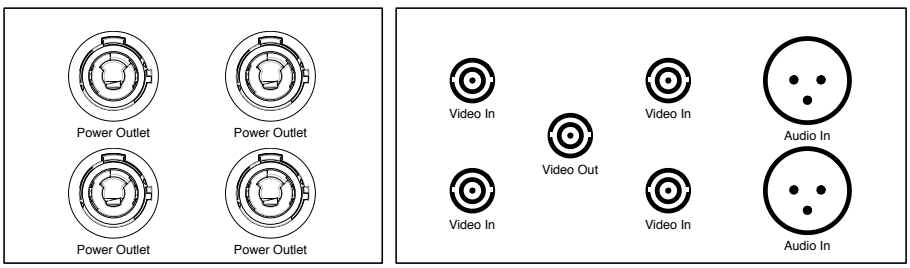

Figure 4-5 - Patch Point Connectors

To conform to the aims of the RME, the connection of ad-hoc cameras and microphones needs to be facilitated in a rapid manner. This will be done using the proposed patch points shown in Figure 4-5. The audio-visual connections consist of 5 BNC video connectors and 2 XLR audio connectors. Also included are a number of low voltage power outlets. These power outlets provide the dedicated 24v AC power supply that drives the static camera circuit. This dedicated power supply provides a connected camera with the ability to synchronise with other equipment installed within the RME.

Figure 4-6 shows the locations of the patch points within the physical space. These positions were chosen as they give an even distribution while avoiding immovable objects such as windows. Two important physical features of the space have also been included in Figure 4-6: a moveable wall and a control room.

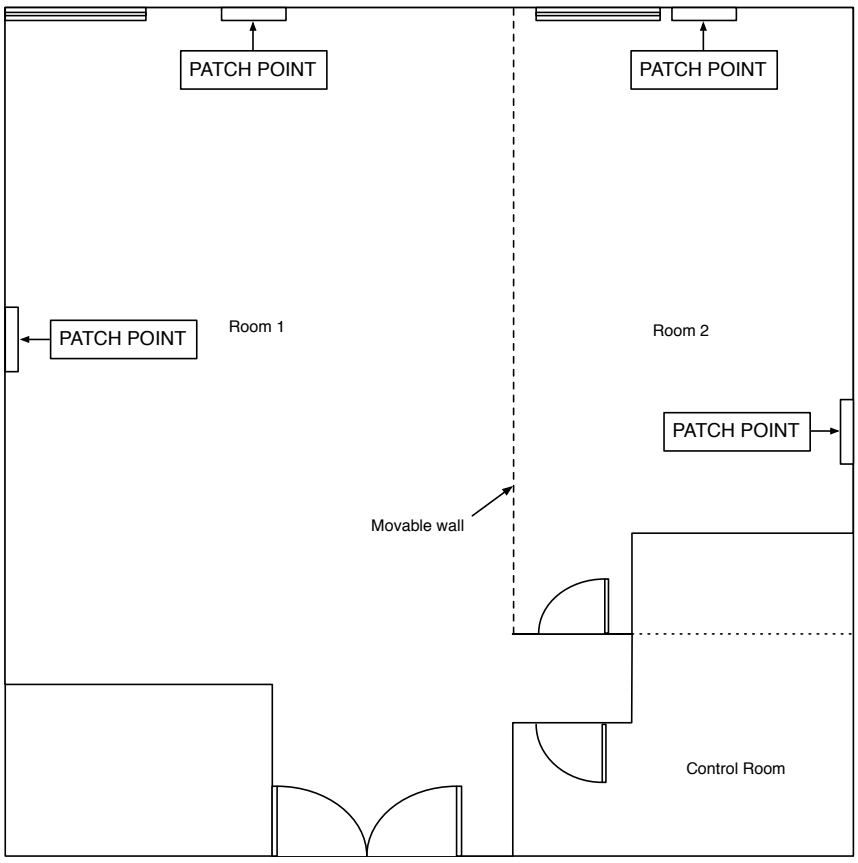

Figure 4-6 - Patch Point Locations

The control room is included due to the need to have a separate area to locate significant amounts of noisy, distracting and valuable equipment. It will also host the wizard and monitoring equipment required for evaluation scenarios. The addition of a moveable wall further enhances the configurability of the space. When in place, it will be possible to run tandem evaluation scenarios. It will also provide users with a physical barrier for simulations that require obscurity. For example, a scenario that involves controlling a device located in another room. Conversely, the barrier is not soundproof, meaning contextual experiments involving sound can also accommodated.

All audio and video feeds will route back to the control room, shown in the bottom right of Figure 4-6. From here a human controller, or wizard, can view up-to 10 video feeds on the preview monitors. The dedicated recording solution will also be located in the control room. This will provide a means of storing captured video and audio footage for post analysis.

4.2.1 Wireless Smartphone Screen Capture

The control room will also host the equipment required to wirelessly capture a smartphone screen. This will provide a direct link between the phone and observation/capture system allowing a wizard to see a prototype in use while directly viewing interactions and gestures performed by the subject on the preview monitors located in the control room.

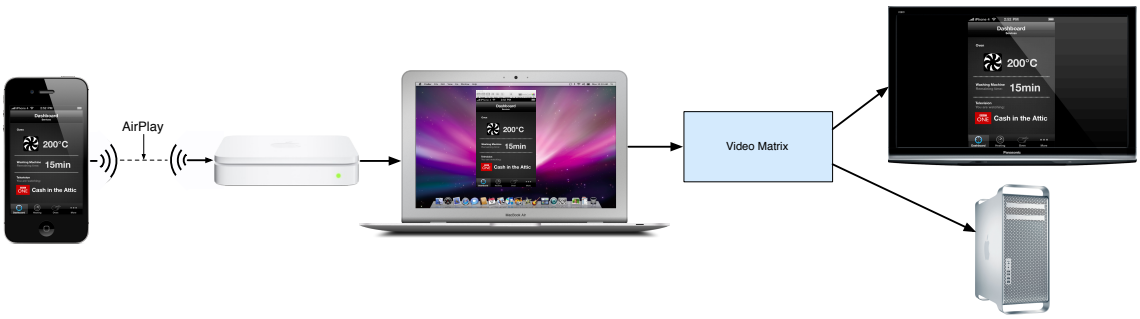

Figure 4-7 - Overview of Smartphone Screen Capture

The solution is based on a technology called *AirPlay* for iOS [160]. This technology was originally developed to allow music and movies, stored on an iOS device, to be streamed wirelessly to a set of speakers or a television. The technology can, however, be configured to send other video feeds wirelessly. Instead of streaming movies it will be configured to stream a mirror image of the smartphone's screen. The smartphone will be connected to a wireless local network within the RME using the 802.11n Wi-Fi standard. From tests, it was shown that older standards (e.g. 802.11g) do not offer the technology required to transmit a real-time video stream. Using a dedicated 3rd party desktop application running within the control room, the video feed will be received on the desktop computer and, using the second display output of the desktop computer, will be connected to the RME observation and capture system. To view interactions and gestures performed by the user, a dedicated iPhone App will be produced that will overlay touch events being received by the phone (this is discussed further in section 6.7). An example of the type of overlays is shown in Figure 4-8.

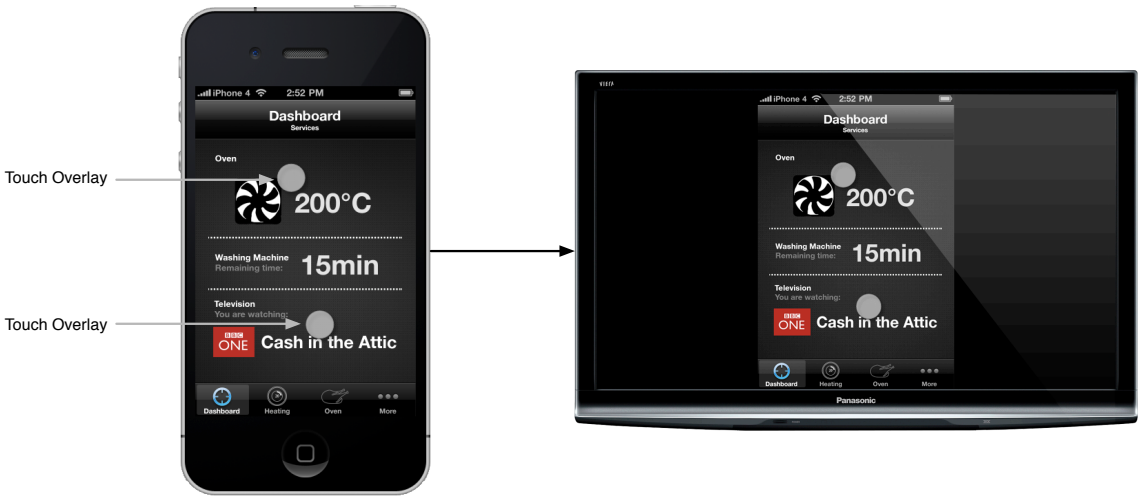

Figure 4-8 - Smartphone Screen Capture with Touch Overlays

4.2.2 Rapid Configurability

A specific aim of the RME is the ability to rapidly configure the space. Currently the routing of video and audio feeds requires physical cables to be disconnected and reconnected. Physical routing requires the user has an understanding of the underlying

hardware installed within the environment. For example, using the traditional routing paradigm, the user would first obtain an input source, with corresponding patch number, and an output source, with corresponding patch number. A physical cable would then be used to connect the two devices from a panel of connections. Configuring many devices is a time consuming and complex process that does not promote rapid configurability. A different approach is therefore required.

For audio, an industry standard audio mixer will be sufficient for connection, and subsequent selection, of multiple audio sources to multiple outputs (the outputs being the space itself, the capture system and a user monitor). The only stipulation is that the mixer requires at least 5 audio outputs and 10 audio inputs. This is based on the need to send 1 audio channel to speakers in the space, record 2 output audio channels, monitor 2 output audio channels and connect the specified number of microphones available in the space.

For video, a matrix will provide the type of instant routing required. A video matrix is often used in a television studio to enable the constant physical connection of video feeds, but the virtualisation of routing. A matrix is a specific piece of hardware with a set number of inputs and outputs that can be connected together in various configurations (shown in Figure 4-9). Using control commands, this configuration can be modified instantaneously allowing any input to be routed to any output.

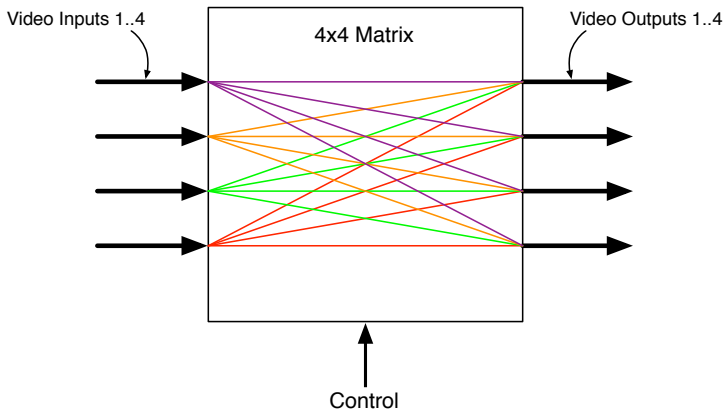

Figure 4-9 - 4x4 Matrix

Based on the number of input and output feeds envisaged and on the manufacturer's specifications, a 32x32 channel video matrix is required. Figure 4-10 shows the proposed routing system for the RME.

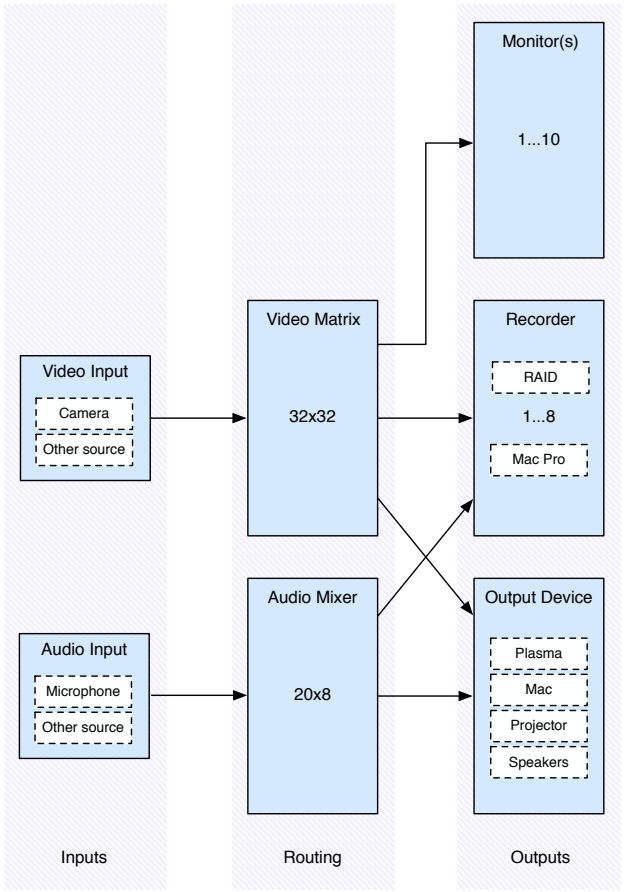

Figure 4-10 - RME Routing System Overview

The use of a video matrix will remove the need for any knowledge of the underlying hardware while allowing for instantaneous routing of video feeds. It does not yet, however, satisfy the aim of rapid configurability because unlike an audio mixer, which includes a visual real-time control mechanism, a video matrix requires an abstract set of output and corresponding input numbers for it to be configured.

Figure 4-11 shows the type of control interface shipped with a video matrix. Although it does allow for instantaneous routing, it is reliant on an in-depth knowledge on number

subsystems, input and output ports and the concept of where a physical location is, based on a universal number. The need for this knowledge is due to the use of video matrixes within broadcast television. Within this type of environment it is appropriate because it is being controlled by a dedicated engineer who will operate it based on experience. The requirement for this low-level engineering knowledge it not suited to a high-level environment such as the RME. A new approach is therefore required.

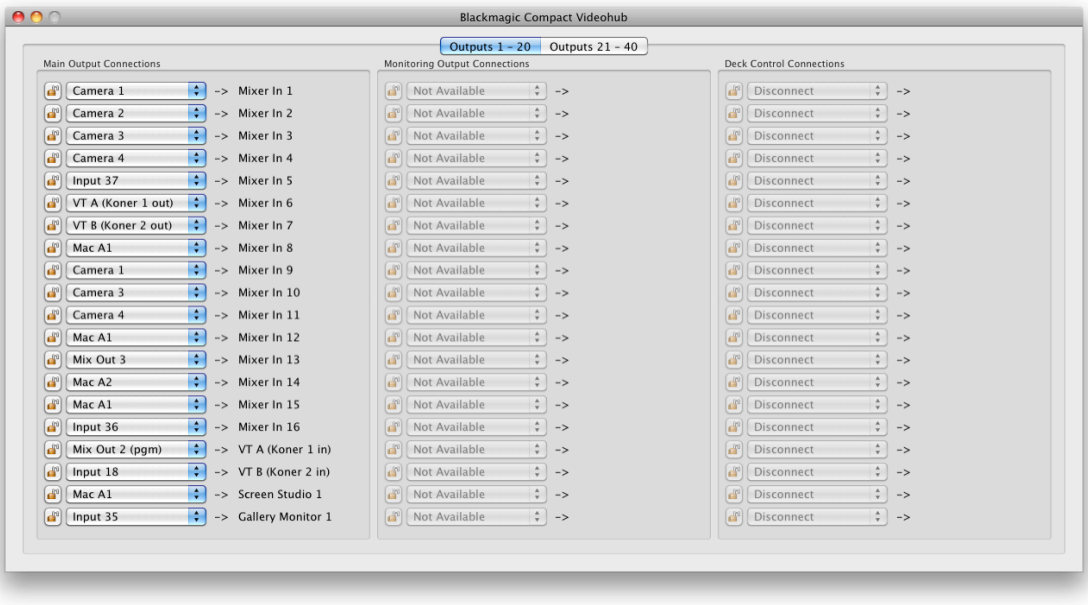

Figure 4-11 - Black Magic Video Matrix Controller

Figure 4-12 shows a system-level schema for the RME with the addition of a control mechanism for the video matrix. The control mechanism is a software layer enabling the visual routing of video feeds within the RME. It is introduced as Visual Matrix.

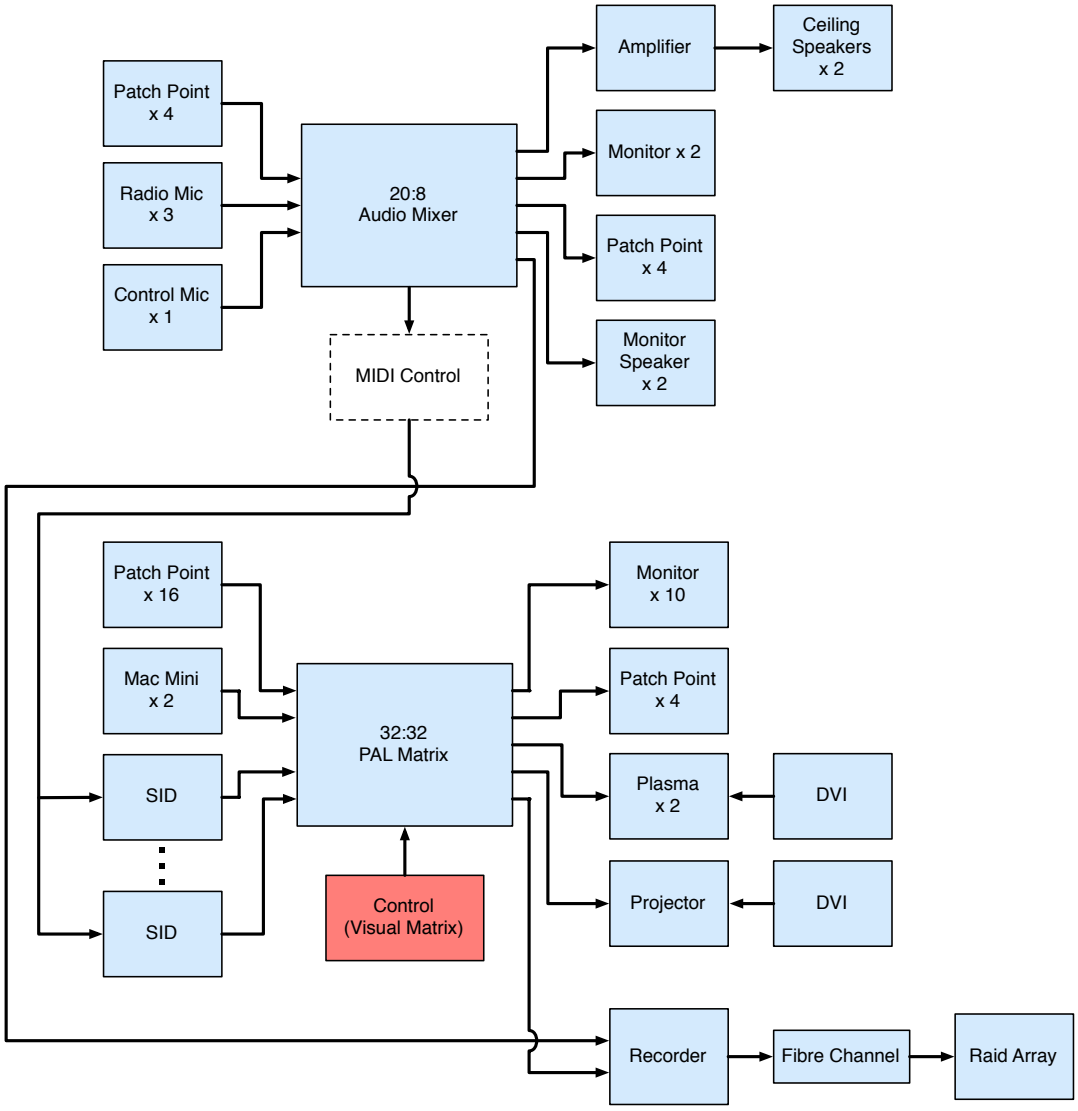

Figure 4-12 - RME System-level Schema

4.3 Control software - Visual Matrix

The Visual Matrix software layer enables high-level routing of video feeds using a unique and novel GUI. With this, a wizard has complete control over all the video signals within the RME. This includes camera inputs, computer inputs, video projectors and plasma screen outputs. In addition to these video devices, any device with a PAL input or output can also be connected to the system. Upon connecting a device, a user will create a virtual representation of the video device. By selecting which physical

connection this virtual device should control, the video feed can be routed between devices using only the software layer. The aim is to allow the user to create a virtual representation of the entire space so they can easily visualise where cameras are and which one is best for covering a particular event. Using Visual Matrix, a user is able to connect input devices to output devices by simply clicking on the virtual representation of the input device and dragging a connection to an output device. A mock-up of this concept can be seen in Figure 4-13.

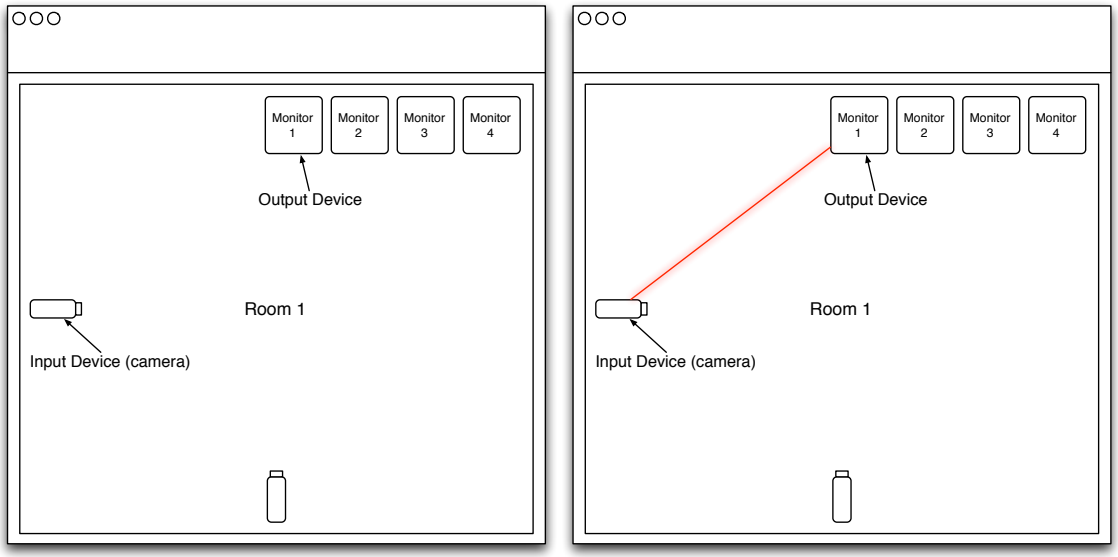

Figure 4-13 - Visual Matrix Routing Concept (Run Mode) Mock-up

It is proposed that the Visual Matrix software layer be designed with two user modes; design mode and run mode. When setting up the lab, the devices (e.g. camera, plasma screen) are physically connected to the pre-installed patch points. When in design mode the user replicates this physical layout by adding various virtual devices to the GUI and moving them into the corresponding place. This lets the user visualise the space on screen, simplifying the task of routing, particularly when happening in real-time. By selecting a device a user can use an inspector panel to specify its settings (e.g. name, type, patch point number). An example for this design can be seen in Figure 4-14.

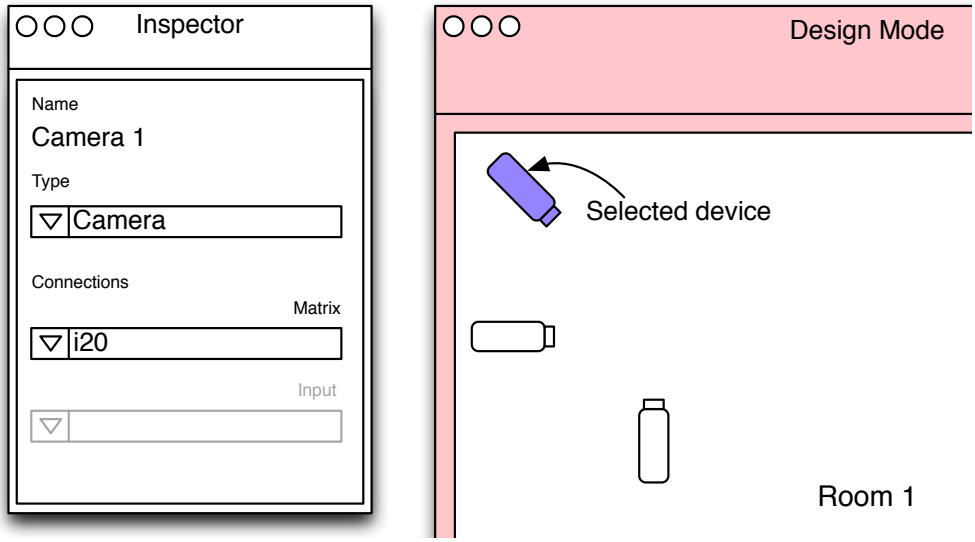

Figure 4-14 - Inspector and Design Mode Example

As the focus of an experiment shifts around the room the user can easily switch cameras to keep the subject in shot, which is handled with run mode. Run mode (shown in Figure 4-13) gives the user freedom to connect any of the input devices to any of the output devices with a simple and fast drag and drop of the mouse. Speed is essential with real-time routing and reconfiguration so being able to visualise the space combined with the natural drag and drop solution means that a user can quickly react to events happing within the environment. Theoretically this could also be achieved automatically using image-processing techniques similar to those outlined by Brumitt et al. [147]. However, due to the size of the space and the rapid nature of evaluations envisioned, it is unlikely that this type of automation will be required.

During evaluation experiments a wizard can react to a subjects interaction within the RME by routing different video feeds to the various monitors within the space. When a connection has been made the Visual Matrix software layer communicates with the video matrix backend and configures the low-level routing. This allows the system to be operated by any wizard with little to no training.

The development of the software will be split into two main areas.

1. Backend, low-level, communication and data management
2. Frontend GUI including controllers required to modify data based on GUI manipulation

Based on available hardware and previous experience, the Apple Mac OS X platform will be used. The need to communicate with the video matrix requires the ability to have low-level operating system access, something that is better provided with a native development environment. Within the Mac OS X environment, Objective-C and the Cocoa software development framework will be used.

Having established the platform for development, an in-depth analysis of the available frameworks and APIs was conducted to better understand the options available to aid development. The result of this analysis presented a clearer picture of what existing tools could be used to improve and speed up the development of the Visual Matrix software layer. Most significantly was the ability to use Core Data. Using the Core Data framework many tasks can be automated using the built in object management provided. This includes tasks such as undo and redo, changes to the data, filtering and organising of data.

Providing Core Data with information about the object model is achieved with a schema that holds a description of the managed objects, or entities, and the relationships between those entities. Using Core Data should reduce the overall amount of code that will need to be developed due to the automatic handlers for functions such as undo and redo. Furthermore, Core Data should better manage the underlying data used within the Visual Matrix software layer thus preventing data from becoming out of sync.

Having a greater understanding of the development environment aided with the production of block diagrams, as well as a managed object model schema for use with Core Data, using the unified modelling language (UML).

These diagrams outline the classes, and relevant relationships required in order to develop and fully implement the Visual Matrix software layer using Cocoa and Mac OS X.

A block diagram for the communication block is shown in Figure 4-15.

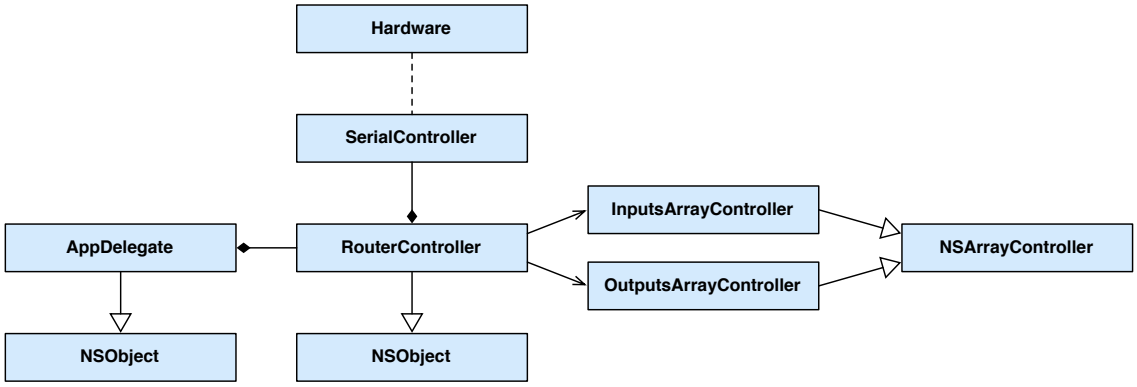

Figure 4-15 - Communication Block Diagram

A block diagram of the Visual Matrix GUI can be seen in Figure 4-16.

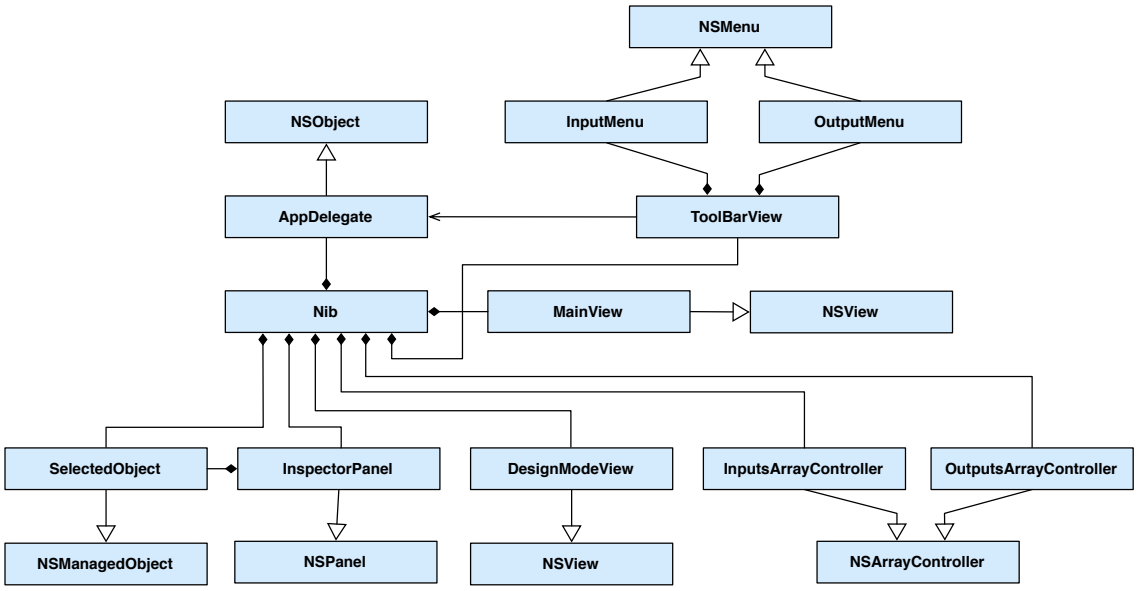

Figure 4-16 - GUI Block Diagram

The managed object model schema is relatively simple as object types can be split into either an Input or Output object. Data held by the object will be used to store information about the virtual object's display settings for the GUI. A Physical Connection Reference will also be stored indicating the physical location where the input or output device is plugged in within the RME. A one-to-many relationship between the objects has also been formed indicating that one input can have many outputs. Using this relationship, arrays can be automatically generated containing either a connected input, or connected outputs, for a specific object. The managed object model schema is shown in Figure 4-17.

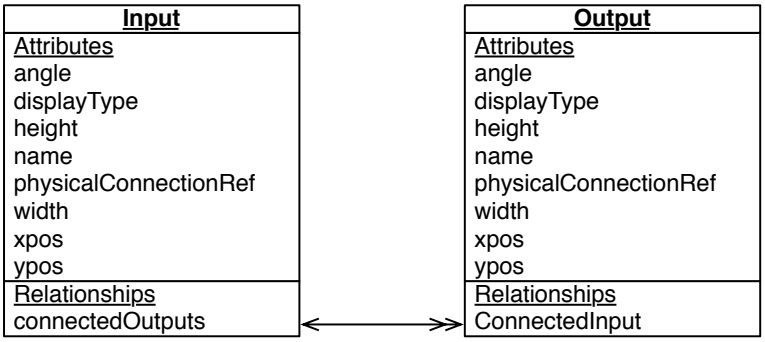

Figure 4-17 - Visual Matrix Managed Object Model Schema

Using Core Data much of the backend data management will be handled automatically using the Managed Object Context (MOC) and Persistent Store Coordinator. Changes to the data will be managed via an *InspectorView* class with the majority of the work handed off to Cocoa bindings. Cocoa bindings is a collection of technologies that can be used within Objective-C applications to fully implement a Model-View-Controller paradigm where models encapsulate application data, views display and edit that data, and controllers mediate between the two [161].

Using the Model-View-Controller Design Pattern employed by Cocoa, shown in Figure 4-18, views can be created using Interface Builder [162], a GUI led development application shipped with the XCode IDE for developing application interfaces. With the

use of Cocoa bindings, views can then be connected to the data (in our case the MOC) using a controller to mediate the two.

Figure 4-18 - Model-View-Controller Design Pattern [163]

Using this model most of the glue code can be eliminated thereby reducing overall development time and the number of potential coding errors. The development of *InspectorView* uses this model, along with Cocoa bindings. Although the use of Cocoa Bindings helps reduce code, it does not eliminate it completely. The creation and deletion of virtual devices is still handled in code within the *AppDelegate,* and certain GUI elements within the *InspectorView* still require significant code to function. For example selecting the type and physical connection reference for a virtual device object is accomplished using coded NSPopupButtons.

4.3.1 Communication

The Visual Matrix software layer uses American Standard Code for Information Interchange (ASCII) characters transmitted using Recommended Standard 232 (RS-232) enabling instantaneous digital control.

This ASCII RS232 data stream enables communication between the Visual Matrix software layer and the video matrix hardware. Control between Visual Matrix and the video matrix hardware is achieved with the use of a number of set commands. The simplest possible command string would be **!!* that consists of the leader and trailer characters but no commands between them. The command would generate the response **OK!!<CR>. The command W1 is used to request that a connection be made. Using the W1 command followed by a comma-separated list of input numbers, one for each output, configures the matrix.

```
**W1,13,4,17!!
```

Figure 4-19 - Matrix W1 Command

Figure 4-19 shows an example W1 routing command. The command will route input 13 to output 1, input 4 to output 2 and input 17 to output 3. The position of the input number will determine the output that it is routed to. The example shown in Figure 4-20 includes 12 comma separated numbers; the 12th number in the list defines which input is connected to output 12, in this instance, input 3.

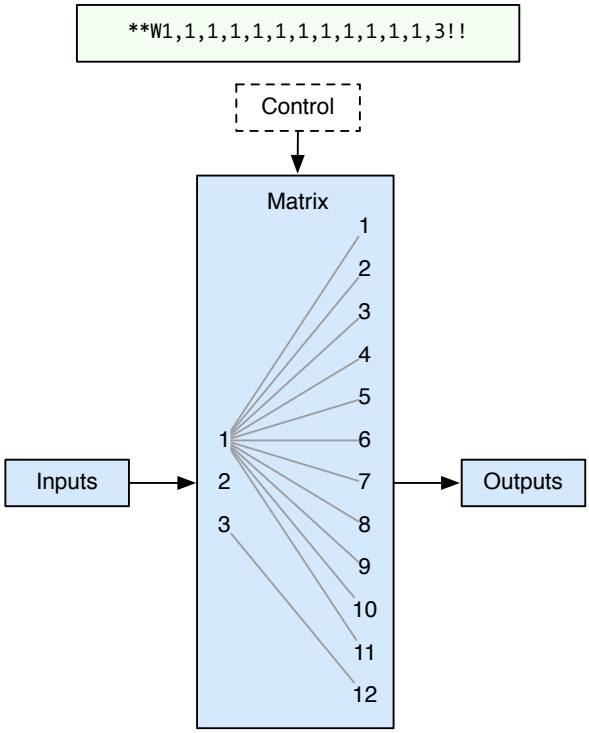

Figure 4-20 - Matrix W1 Command Output

Other commands such as **S!! (used to request the status of the matrix) and **CLEAR!! (used to clear all current connections) can also be used to add extra functionality and control.

The formation of command strings is handled by the *RouterController*, utilising the *InputsArrayController* and *OutputsArrayController* and passed to the *SerialController* for delivery to the router hardware.

Due to the lack of serial RS-232 ports available on modern computers an USB-RS232 converter is used with the *SerialController* responsible for the low-level communication and handshaking between the application and converter (this process is shown in Figure 4-21). The *SerialController* is a C class originally written by the USB-RS232 manufacturer and modified to better fit within the Visual Matrix software layer. As well as being responsible for the converter handshaking, it is used to send and receive command messages to/from the router.

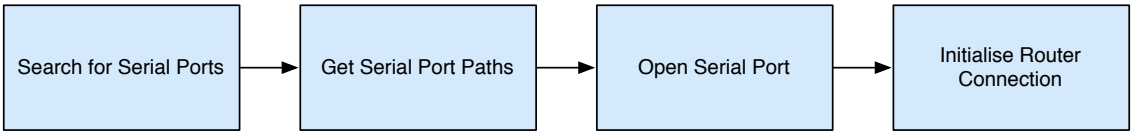

Figure 4-21 - USB-RS232 Initialising Process

4.3.2 Frontend Graphical User Interface

The Visual Matrix GUI is split into three main areas, the main content view, toolbar and a separate inspector panel. Using these three GUI components a user can build a virtual representation of the RME for high-level video routing. The general layout of the GUI can be seen in Figure 4-22.

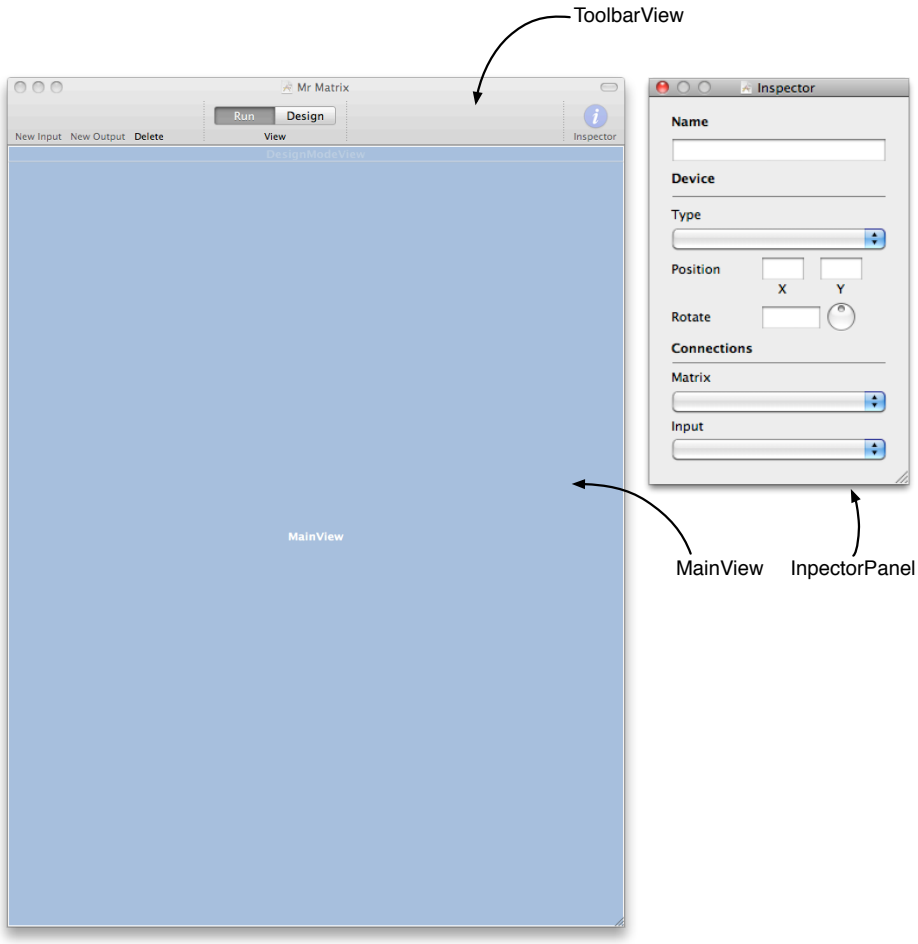

Figure 4-22 - Interface Builder Output

Initial GUI development used Interface Builder with the output of a *nib* file containing interface objects. Nib files can be loaded from within an application with the containing objects being initialised automatically at runtime. Connections can be made from objects within the application and objects stored within Interface Builder. This makes interface object creation and management more efficient. The objects stored within the Visual Matrix nib file and an example connection can be seen in Figure 4-23.

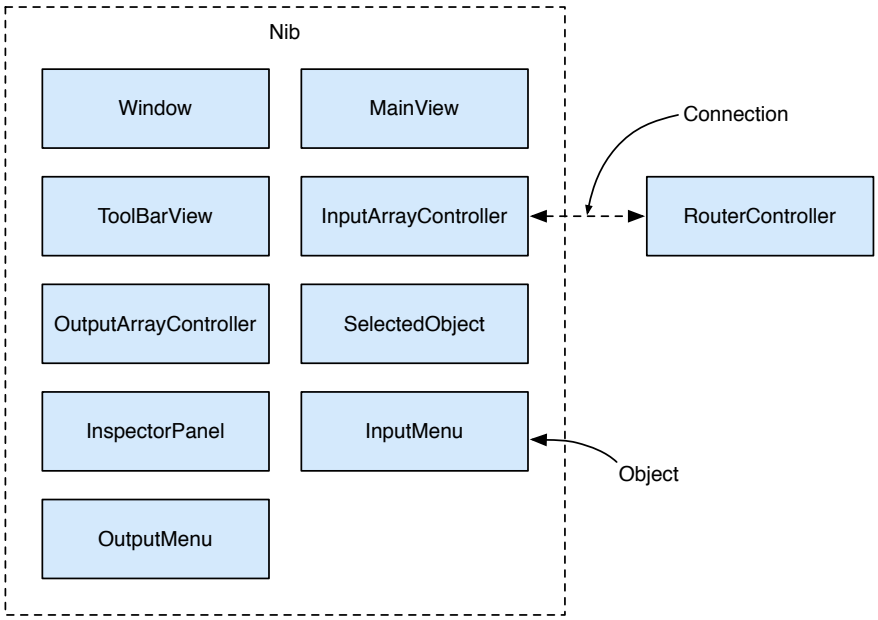

Figure 4-23 - Visual Matrix Nib Objects and Example Connection

Interface Builder quickly enabled the basic structure of the GUI to be laid out, especially the *InspectorPanel,* as this mainly uses Cocoa Bindings to link the backend data to the frontend views elements. However, the *ToolbarView* and *MainView* can be considered shells as most of the functionality comes from source code. The Interface Builder output can be seen in Figure 4-22.

The *MainView* uses a full re-draw mechanism when any change in the Managed Object Context occurs. Using a notification event sent from the either the *InputArrayController* or *OutputArrayController* the drawRect method within the *MainView* gets called. This

in turn brings all the objects from within the Managed Object Context and renders them on screen (shown in Figure 4-24).

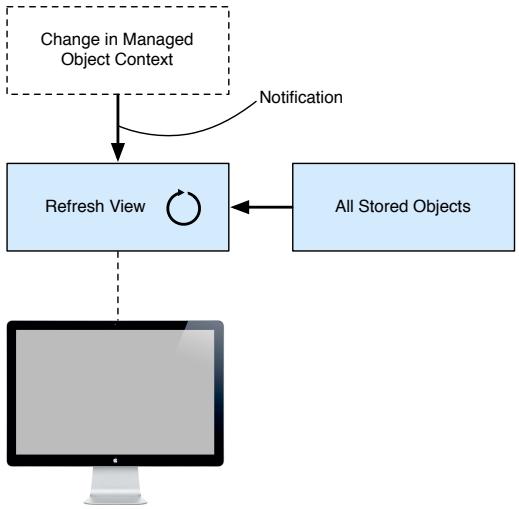

Figure 4-24 - Visual Matrix *MainView* Refresh Mechanism

The variables stored within either the input or output managed object determines the position, type and angle with which the image (the virtual representation of the input or output device) is drawn on screen (as seen in Figure 4-25). This is achieved by iterating over every object within the MOC, extracting the relevant information, and using it to draw an image.

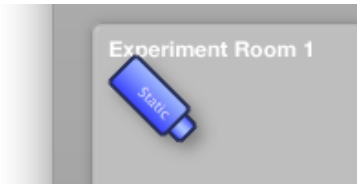

Figure 4-25 - Visual Matrix Drawing Virtual Representation of Physical Device

Visual Matrix can be used in two distinct modes; run mode and design mode. Virtual objects respond differently to a manipulation depending on whether the software is in

run or design mode. When in design mode the user can move an object around within the *MainView* thereby adjusting the x and y coordinates of that object (other variables such as type are set using the *InspectorPanel*). The x and y coordinates are set using the mouse down, drag and mouse up events. When in run mode clicking on a object will allow the user to make a connection between that object and another object within the *MainView* (thereby physically connecting two devices within the RME). The Visual Matrix mouse events state machine is shown in Figure 4-26.

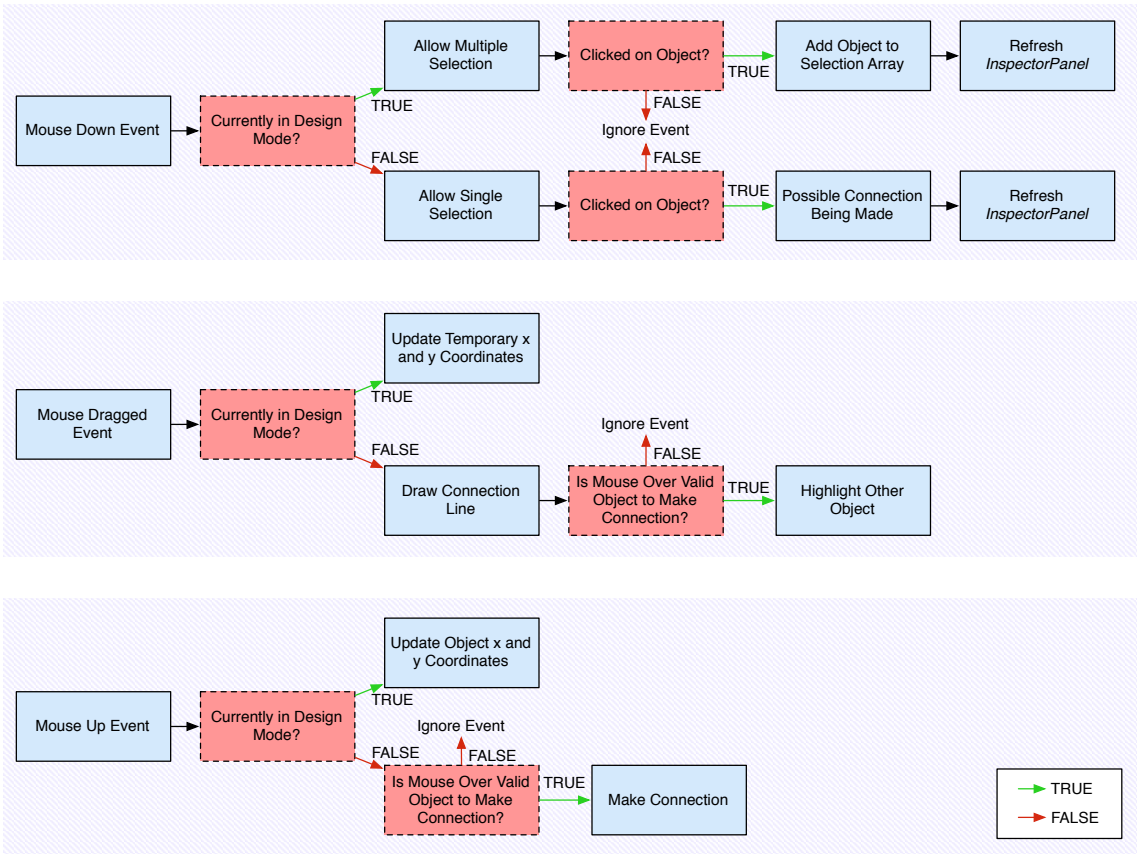

Figure 4-26 - Visual Matrix UML State Machine Diagram (Mealy Machine)

The *MainView* handles other functionality such as snap to grid and multiple selections, while the *ToolbarView* (along with the *AppDelegate*) handles the creation and deletion of the objects within the MOC.

Figure 4-27 shows the final implemented GUI in run mode with the visual routing paradigm in action. Clicking on an input device (in Figure 4-27, a desktop computer) and dragging a connecting line to an output device (in Figure 4-27, a projector) enables instantaneous visual routing of video feeds.

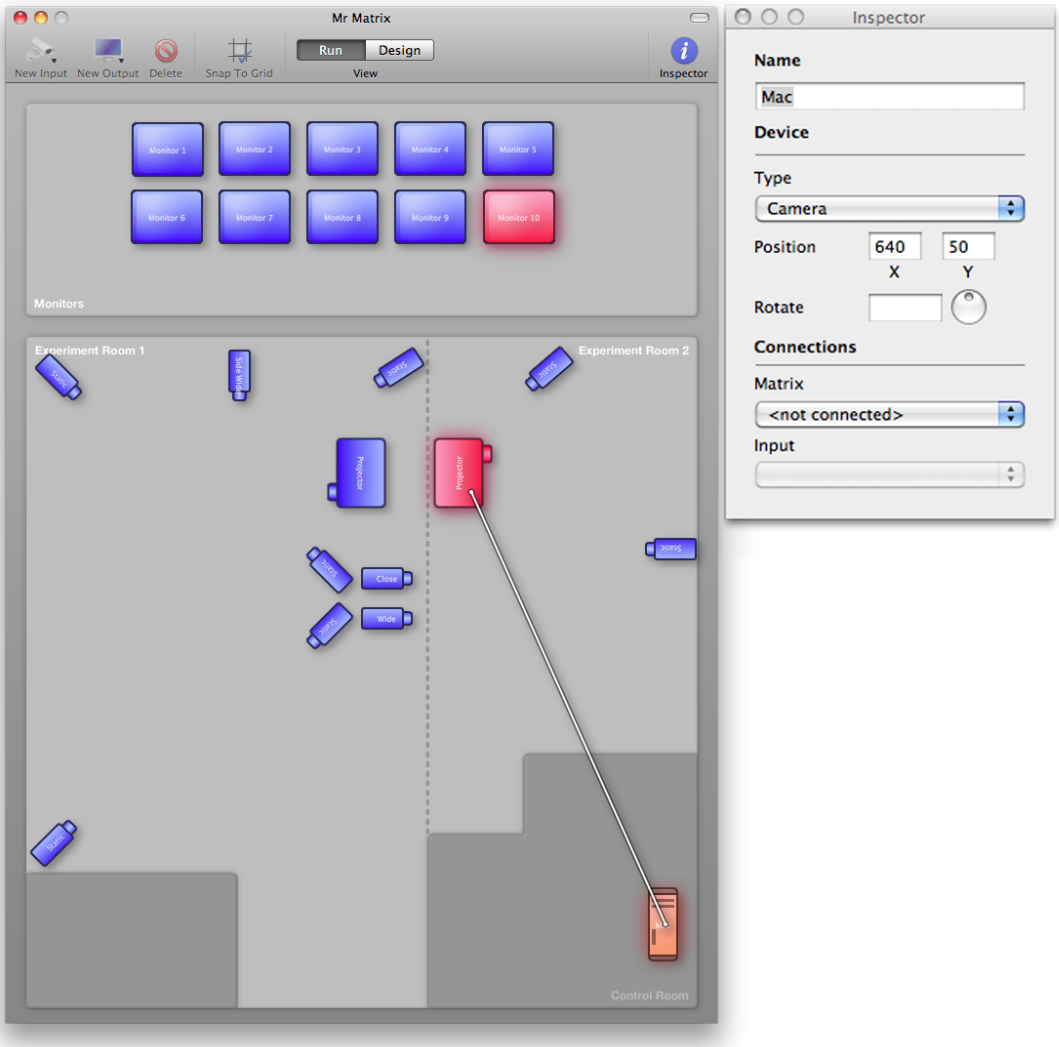

Figure 4-27 - Visual Matrix Final GUI

Figure 4-28 show the physical layout design used for the RME with the actual constructed space. It incorporates numerous cameras, screens, projectors, patch points and computers. With the use of the Visual Matrix software layer a wizard can have instant control over all the feeds being routed from/to devices.

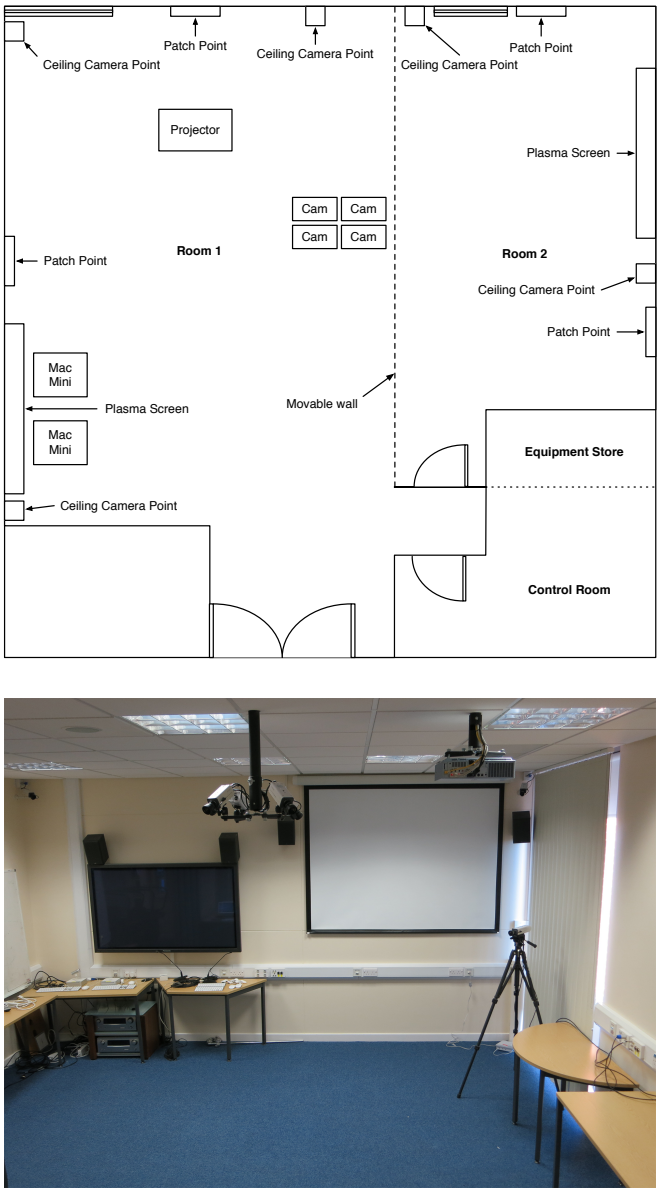

Figure 4-28 - [top] RME Physical Layout Design and [bottom] the RME Evaluation Space

4.4 Summary

This chapter has focused on the design and implementation of the RME. Although similar environments exist, there are none within the field of smart home or ubiquitous computing that enable both the rapid configurability, data capture and a platform for simulating smart home devices (discussed in chapter 5) all aggregated within the same system and physical space. It can therefore be said that the RME is a novel system that enables previously difficult or impossible user centred evaluations to be performed [37, 38].

Empirical research can be facilitated using many methods, which span across a broad array of subject areas. The RME aims to provide a space where this type of research can easily be conducted with all relevant data captured. It does so by considering the current and past literature and building upon already established methods. The configurability of the RME enables a broad range of research areas to utilise the space with a rapid setup and evaluation.

As a standalone unit the RME is highly configurable evaluation space for conducting user centred research. However, it is also designed to provide a platform for simulating smart devices. Using the SID tool, devices can be introduced into the RME thereby enabling interactive experiments to take place in a pseudo realistic smart home environment.

Chapter 5

Simulating Interactive Devices and the Extensible Feedback Mechanism

This chapter outlines the design and implementation of the SID and EFM framework components. The SID component is a video-based development and control tool for simulating interactive devices found within a smart home. The Extensible Feedback Mechanism (EFM) is a controlled two-way visual communication tool. Both components are software layers built on top of the RME physical hardware layer. Figure 5-1 shows an example SID in use within the RME control room [37, 38].

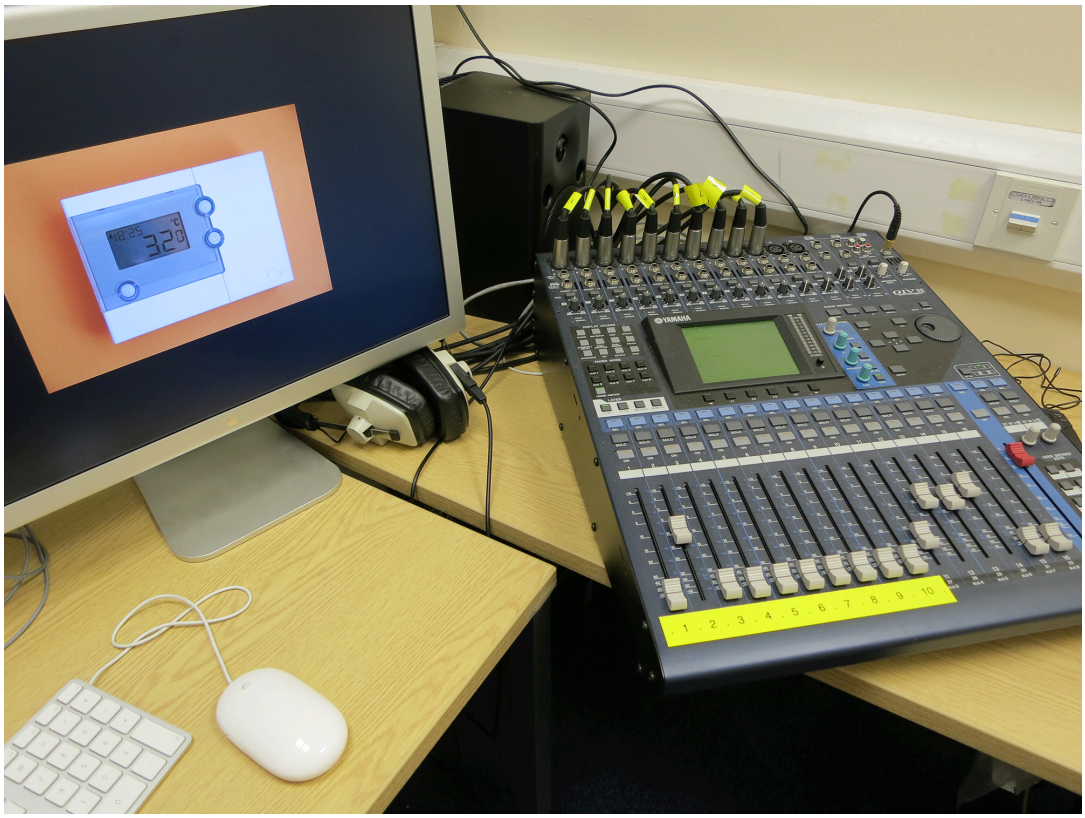

Figure 5-1 - SID in use within the RME Control Room

The SID tool will be split into in two distinct elements: a SID video player and interchangeable SID macro patches. An overview of the SID tool is shown in Figure 5-2.

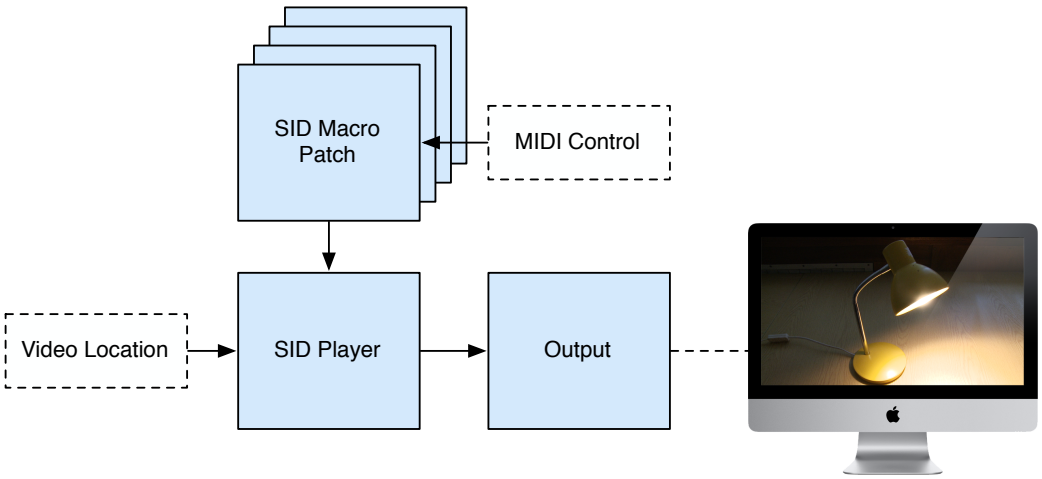

Figure 5-2 - Overview of SID Tool

5.1 Simulated Interactive Device (SID)

The SID tool is a means of providing a simulated version of devices normally found within a domestic environment, for interaction between a subject and a prototype interface. These devices will aid in the evaluation of prototype interfaces for smart home control. Based on the review outlined in section 2.6.6, it is proposed that at the core of the SID tool will be a real-time compositing engine. This will be used for manipulation of video to simulate the functionality of physical interactive devices. Using this manipulation, and the Wizard of Oz technique outlined in section 2.6.3, a wizard can modify the video to make it appear to a subject that real interaction is taking place.

Using the example of a fire, this could now be simulated by recording a real fire in use at various power levels and then compositing looped clips together. Layering the different videos on top of one another and then changing the opacity of each video provides the effect of the fire being off, on or at various power levels. Figure 5-3 illustrates this example and the various fire states. The base layer is the fire in an off state. Mixing a second looped video of the fire at medium power, over the base layer, enables a transition from off to medium power. Mixing a third looped video of the fire at full power provides the final transition from medium power to full. The recent increase in quality of personal camcorders and smartphone cameras mean they can easily provide the required level of quality required for recording these types of assets.

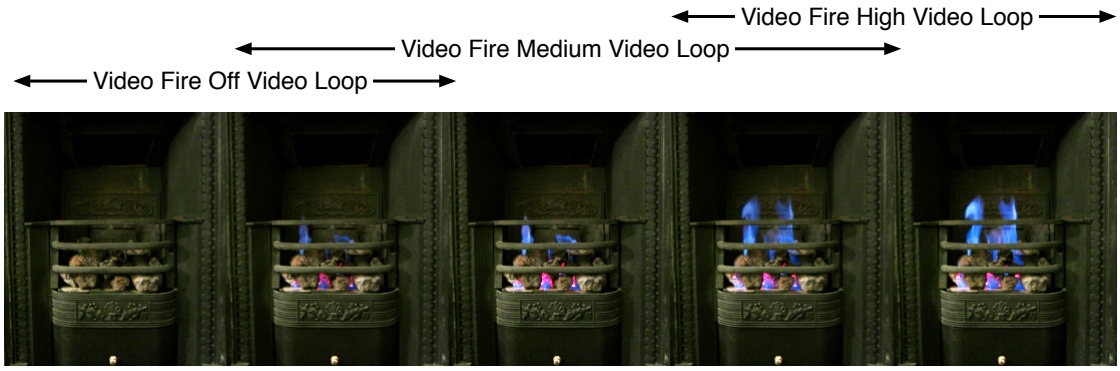

Figure 5-3 - SID Fire Example

Using a large gamut of variables, a designer will have control over many different video characteristics, allowing them to produce a large set of contrasting simulations. The type of variables that might need to be controlled can be taken from those implemented by video effects packages. These parameters include, for example: x,y,z position, x,y,z rotation, width, height, speed and colour. For the effect to be convincing, these variables must be capable of being manipulated in real time. In addition to using captured video and computer-generated (CG) graphics, customised text will also be available for real time composition. This is especially useful when simulating control panels with information displays. Using the SID tool it will be possible to create a whole range of different devices for the subject to interact with.

The basic principle of the SID tool is to composite many simple video elements together in real time, with the result being a complex interactive simulation. The use of video will also allow the SID tool to easily integrate with the RME. Although the current generation of video effects packages already provide this type of video manipulation proposed for the SID tool, there are two key issues which prevent their use:

1. Lack of real-time rendering
2. Lack of a real-time control mechanism

Lack of real-time rendering - Although real-time rendering is not supported by current generation of video effects packages, it is by choice rather than a limitation. Video effects applications are designed to produce guaranteed high quality renders, and because of this they make a trade off with rendering time. Other applications, however, will allow a lower quality or frame rate to enable real-time rendering to occur. Game environments, for example, are designed to be played and rendered in real-time. Using the same technologies as those used in the games industry will provide a means for real-time rendering for SIDs.

Lack of a real-time control mechanism - The lack of a real-time control mechanism is the more difficult issue. As shown in the Figure 5-4 the standard method for controlling various video parameters is via mouse and keyboard centred GUI. Although applications can be controlled in different ways, e.g. timeline-based compared to node-

based, the basic principle is that the user will only want to control a single attribute at a time.

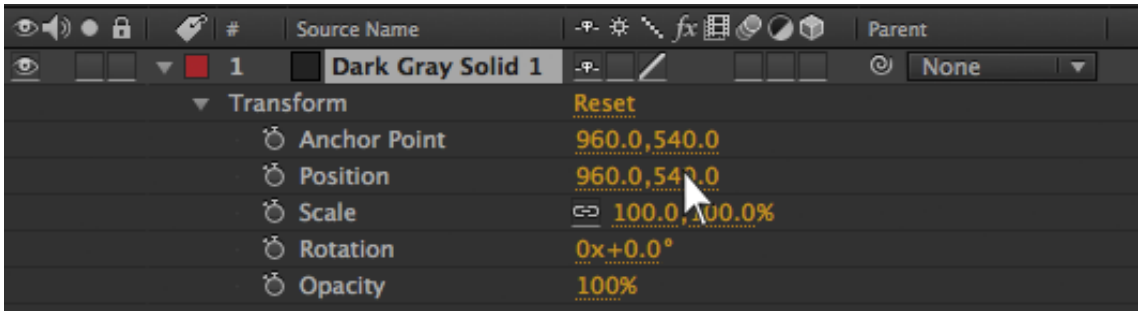

Figure 5-4 - Adobe After Effects Control Mechanism

This type of control is evident across all desktop GUI video applications. The SID tool requires a parallel real-time control mechanism due to the potential need to change multiple video attributes across different SID outputs, all at the same time. A new approach is therefore required.

As shown in Figure 5-5, tablets can provide a novel interface for parallel control of devices. However, they currently suffer from the problem that they do not offer any tangible feedback to the user. Without this feedback a user cannot accurately know the position of a control without looking at it. In a space such as the RME this would be extremely difficult for a wizard, as responses are based on user interaction. If a wizard is required to look at the interface they are controlling, it is difficult for them to view the current scenario. Tablets therefore offer a poor control mechanism for the type of control required in the RME.

Figure 5-5 - Fusing an Audio Mixer Control Paradigm with an iPad

As outlined in section 4.2 the RME will make use of an audio mixer for real-time audio routing and control (as shown in Figure 5-6). The mixer has been designed for parallel control of audio, but could the same control paradigm be used for control of any variable? If so, the wizard could use the same tangible interface for controlling many aspects of both the RME and SID.

Figure 5-6 - Yamaha Digital Audio Mixer [164]

By focusing on the abilities of a traditional audio mixer in providing a real-time parallel control, the same control interface could be used for controlling both the audio and SID devices in real-time. It is proposed the audio mixer is configured to produce Musical Instrument Digital Interface (MIDI) control signals, in addition to controlling audio signals. MIDI is an industry standard protocol originally designed to allow digital electronic instruments to communicate with other electronic devices including computers. Using events messages MIDI transmits musical information such as musical notation, pitch and velocity as well as control signals such as volume, sustain and panning.

Although originally designed for musical instruments, MIDI signals can be used to control anything that implements the protocol. A keyboard, for example, will use a volume control signal to adjust the main volume of an instrument. Instead of adjusting volume, however, this control signal can be used to control other variables. In the case of the SID tool it can be used, for example, to adjust the opacity of a video image. Setting the fader to 0% will produce a transparent image, increasing it to 100% will produce an opaque image. Using many different control signals, each connected to a different fader, will allow a wizard to manipulate different parameters of a SID in real time. An example of this type of configuration can be seen in Figure 5-7.

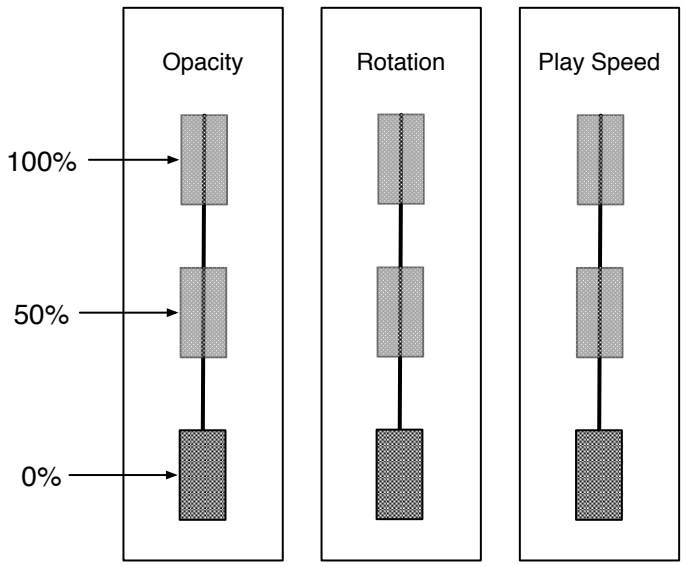

Figure 5-7 - Configuring Digital Audio Mixer To Control Video

With the use of tangible controls such as those provided by an audio mixer, the wizard can look at the scenario being played out within the RME while controlling SIDs. Its also provides a means of parallel control whereby many faders can be changed all at the same time. The use of a single control device will also further reduce the strain placed upon the wizard. The audio can be configured to control both SIDs and audio from the same control panel.

5.1.1 Quartz Composer

An industry standard specification for high performance graphics is the Open Graphics Library (OpenGL) [165]. It is employed widely within the games industry for rendering graphics in real-time. Within the Mac OS X environment the NSOpenGL API layer, built on top of the standard OpenGL layer, provides better integration within the application workspace. This API will provide the core processing functionality required for the SID tool.

A study of the best tools and development aids was completed that resulted in the use of Quartz Composer as a development platform, due to its integration with the OS and visual development style. "*Quartz Composer is a visual programming environment…that allows you to quickly create sophisticated motion graphics compositions without having to write code. By simply connecting together building blocks of graphics processing functionality, you can rapidly design dynamic visualizations that, for example, combine images and real-time information over video feeds"* [166].

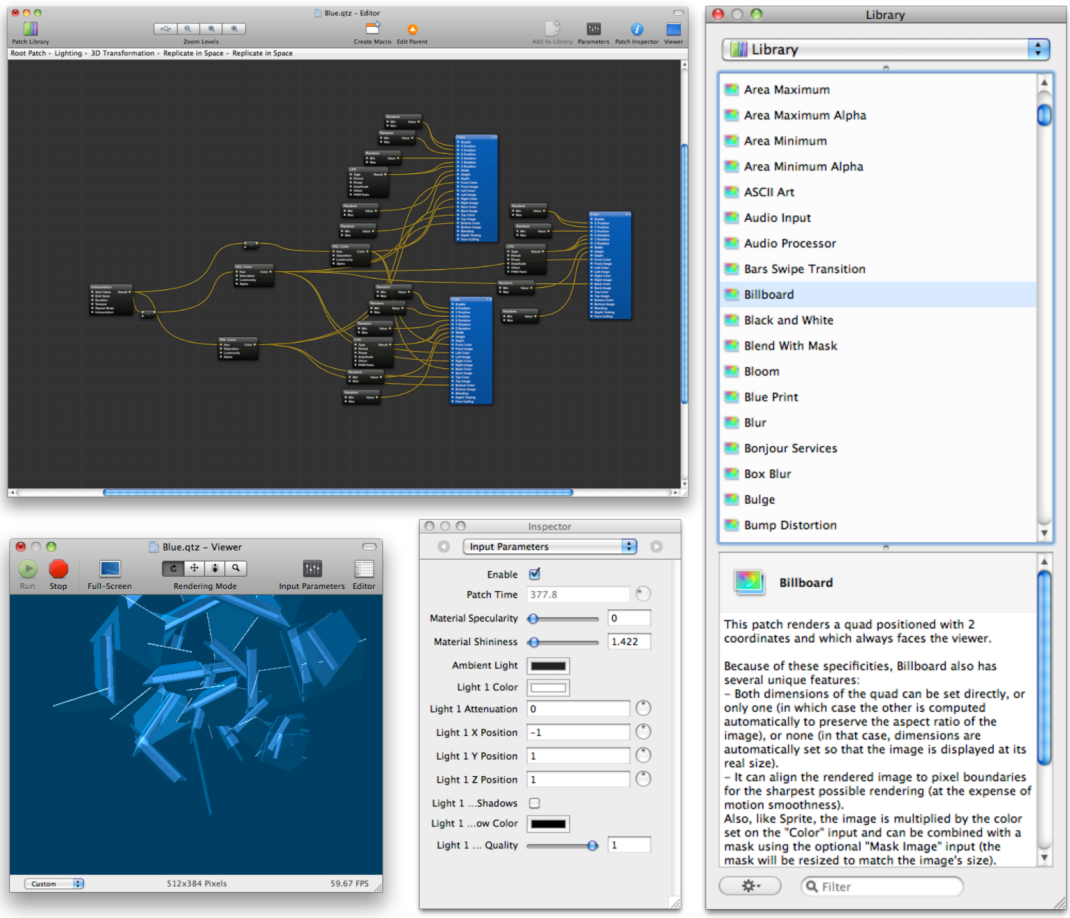

Figure 5-8 - Quartz Composer Example. Windows Clockwise From Top Left: Editor, Patch Library, Inspector and Viewer

Quartz Composer utilises existing components from a library allowing a designer to instantiate so called patches, each of which perform a particular task, and connect them together using a visual connection process. Using various combinations of patches, complex graphics processes (usually involving large amounts of code) can be reduced to a simple drag of the mouse. A collection of patches can be grouped together thereby creating a macro patch (each containing any number of child patches), which allows the underlying processes to be hidden from a user in a black box process. An example of a macro patch would be a seven-segment LCD. This could be dropped into a simulation and controlled via the audio mixer. The patches being used in the editor, shown in Figure 5-8, are in fact child patches of the higher macro patch and have been expanded

for the purposes of the example. This hierarchical structure (shown in Figure 5-9) is extremely powerful, enabling the re-use of the patches in many different packages.

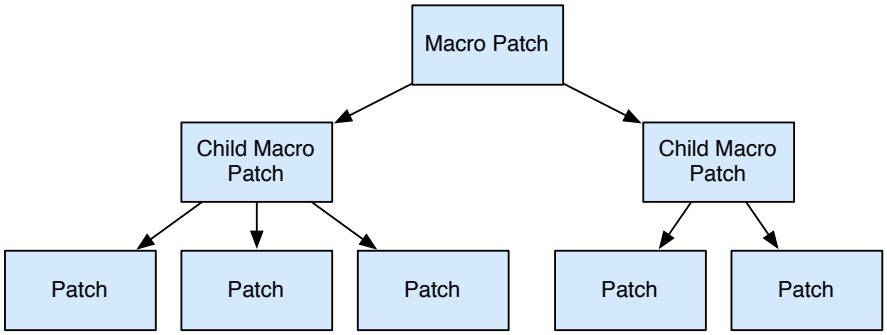

Figure 5-9 - Quartz Composer Hierarchical Structure

Using this hierarchical patch structure, components can be developed for the SID tool for use by designers who are not proficient with coding. A black box veil can hide the complex logic required for video manipulation providing designers with a simpler approach to development. It also enables easy re-use of video manipulation processes.

A SID macro patch will define the type of manipulation applied to an input asset when MIDI input signals are received (e.g. increase opacity or rotate x axis). Many macro patches can be built to give a designer an array of options for creating different simulated devices. If one patch does not have the desired effect then a different patch can be loaded that will. As part of the framework a library of macro patches has been created that make SIDs accessible to designers. These macro patches form the basis of the SID tool and allow basic effects to be easily implemented. It is envisaged, however, that more macro patches will be added to this library as the SID tool is used.

As well as video manipulation the SID tool can be used to generate real-time graphics. Figure 5-10 shows the use of real-time graphics composited with static images to model a thermostat controller. Using a number of bespoke SID macro patches, such as the MIDI Controller and LCD Number, MIDI signals from the audio mixer are fed into the package and are used to control a read out on the thermostat display (in this case the desired temperature and humidity). A background image of a real world thermostat is

used to further increase the validity of the simulation (as discussed in section 2.6). If, during a Wizard Of Oz evaluation session, the subject interacts with the display a wizard can respond in real time, whilst, importantly, also being able to control other elements of simulation at the same time using other faders.

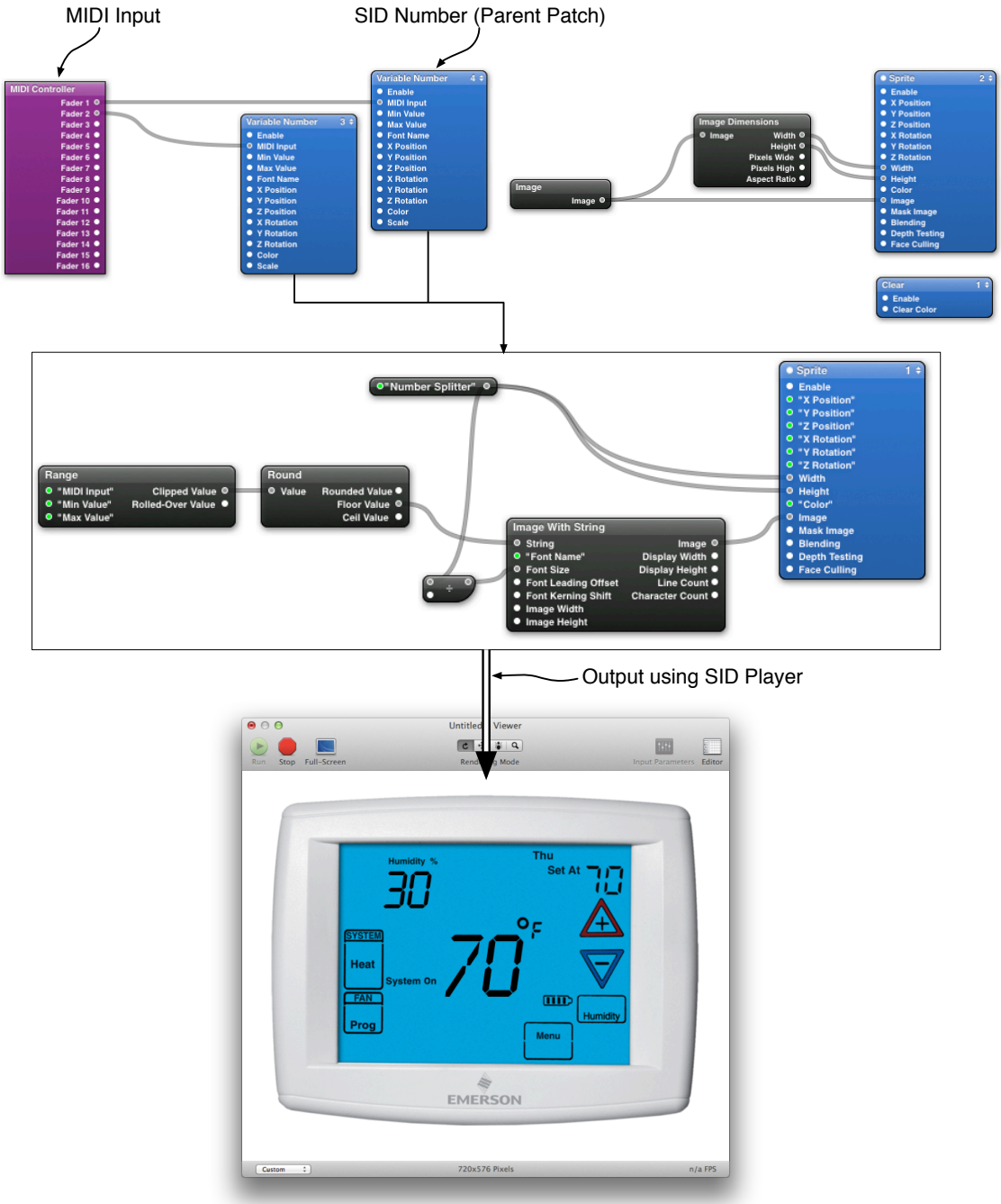

Figure 5-10 - Thermostat Input Package and Output

Figure 5-11 shows the same thermostat output with the individual composited layers highlighted. These independent layers can be each modified with individual faders.

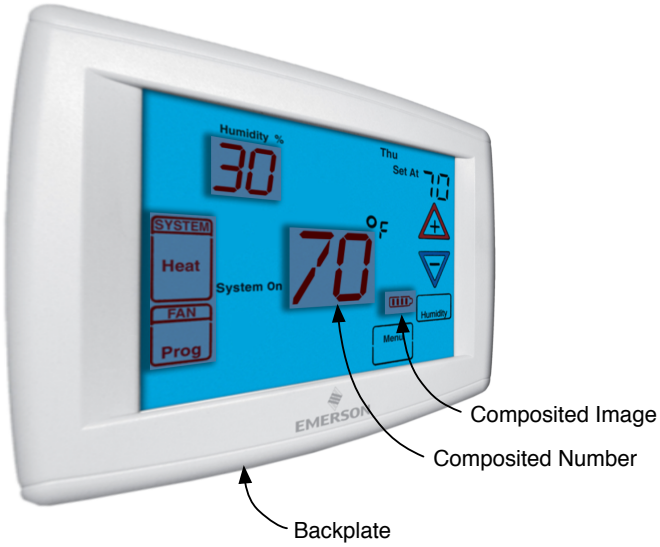

Figure 5-11 - Thermostat Output With Highlight Composited Layers

The output of a package will be pushed to the SID player for rendering and display. The SID player enables full screen rendering of video and Quartz Composer packages within an application. It is used to produce a full-screen render on either a primary or secondary display. At the heart of the SID player is a Quartz Composer rendering engine. This engine takes frames produced by SID macro patches and renders them into a full screen OpenGLContext buffer, using a maximum refresh rate of 50 frames per second (shown in Figure 5-12).

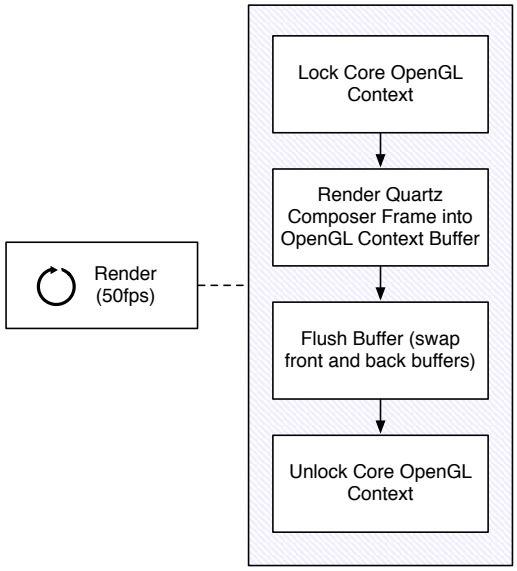

Figure 5-12 - SID Player Render Process

The SID player does not handle any video manipulation; this is handled by the SID macro patches using Quartz Composer and the underlying power of OpenGL. Rather, the SID player is an aggregation application, bringing together the SID macro patches and system settings into a single, easy to use application with the purpose of rendering a final video to an output display [39]. The final output of the SID tool will be a PAL video feed. This will be routed into the RME where it can be viewed using one of the many display outputs installed within the space.

Figure 5-13 shows an overview of the communication model used within the RME with the inclusion of the SID tool. One major benefit of the designed model is the ability for a single wizard to have control over all communication signals used within the RME.

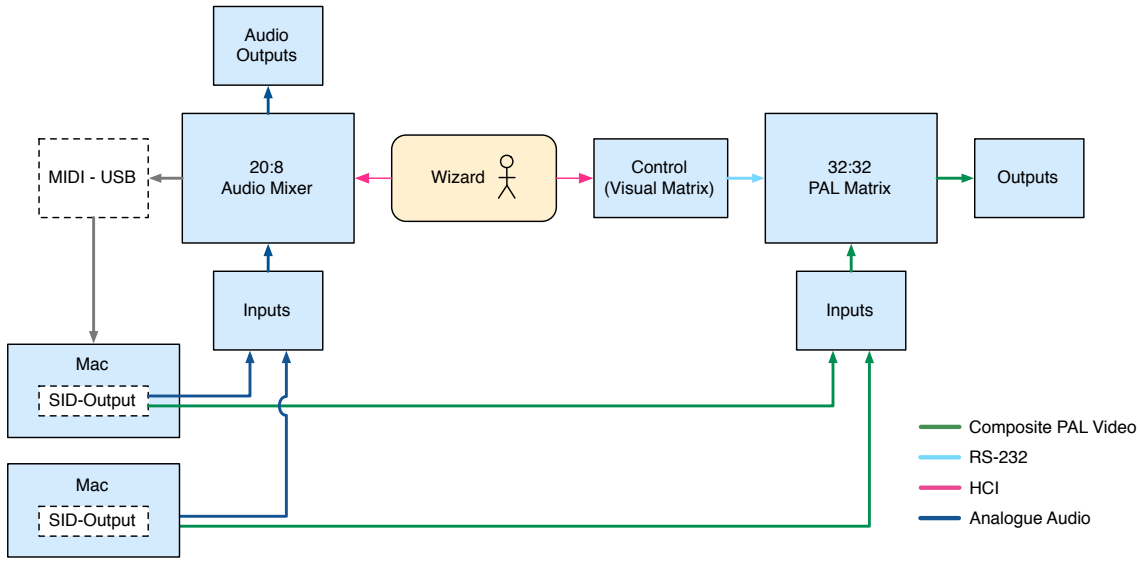

Figure 5-13 - RME/SID Communication Model

The SID tool is a software layer built on top of the RME hardware layer. It can be used to create video simulations of physical devices found within a domestic environment. These simulations can be manipulated in real-time by a wizard allowing for pseudo realistic interaction to take place between a subject and simulated device. Due to the variable and inconsistent nature of user research, the SID tool has been designed, through the use of a library of SID macro patches, to allow a wide variety of devices and configurations to be implemented. Moreover, being based on Quartz Composer allows for, among other things, extensibility through the creation of more macro patches for new research threads. For example, if a SID is required that has yet to be developed, previously made macro patches could be used as the building blocks for the new SID or new macro patches could be developed.

5.2 Extensible Feedback Mechanism (EFM)

The Extensible Feedback Mechanism (EFM) is a tool for WOz studies that involve text based two-way communication between a subject and a computer system or smart device. However, in keeping with the WOz technique and unbeknown to the subject, the communication will in fact be between a subject and a wizard. The wizard will take on the role of the computer system or device. Due to the different requirements of the

subject and the wizard, the type of GUI used by the feedback mechanism will be configurable via plugins. These GUIs are not being evaluated, but rather, they allow communication between a subject and a wizard using different control visualisations that will meet the requirements of each user. The EFM tool will consist of three main components: EFM-Wizard, EFM-Subject and various plugins. These plugins will allow for the production of different GUIs for either the wizard or subject. EFM-Wizard and EFM-Subject are two independent applications, which, when run (and connected via a network), allow plugins to send messages between them (as shown in Figure 5-14).

The real power of the EFM lies with the architecture shown in Figure 5-14. A wizard may require a fundamentally different interface to the subject if they are to respond instantaneously, in a manner similar to a computer system.

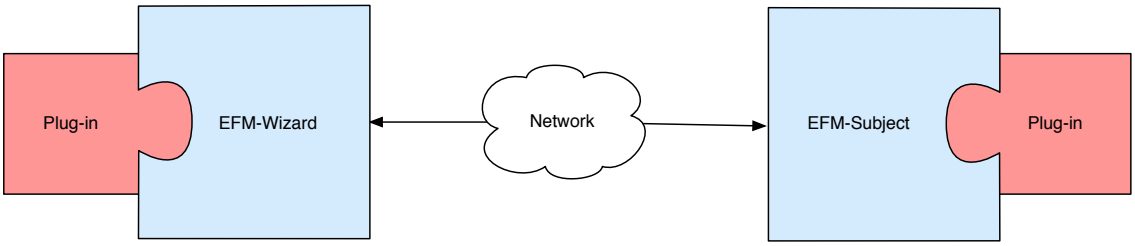

Figure 5-14 - EFM Architecture

Unlike existing applications that have rigid GUIs, the EFM is designed to allow the subject and wizard to communicate through sending and receiving generic messages to/from a plugin. How a plugin handles a message is not defined in the architecture, allowing for any type of GUIs to be *bolted on* to the front end. Using this type of architecture allows for infinite extensibility through the design of new plugins. If a new communication interface is required, one can be produced as a plugin and attached to the host application where access to the common underlying communication protocol can be granted.

As shown in Figure 5-15 the EFM will have two host applications - the main control application, EFM-Wizard and the connection terminal, EFM-Subject. Using EFM-

Wizard a user can connect to, and communicate with, a single instance of EFM-Subject and control the logging of messages. Messages are logged as XML and tagged as either a sent message (therefore a wizard message) or received (therefore a subject message).

EFM-Subject is designed to be similar to a dumb terminal. A subject is able to send messages; however, all other control lies with EFM-Wizard. Using the base application, a wizard can make a connection to an instance of EFM-Subject and once a connection has been made the underling GUI (which consists only of a connection status indicator) will change from *Not Connected* (a red icon), to *Connected* (a green icon). A subject is then able to start sending messages to the connected EFM-Wizard instance using the custom plugin GUI. It is envisaged that this setup process would happen prior to the evaluation session.

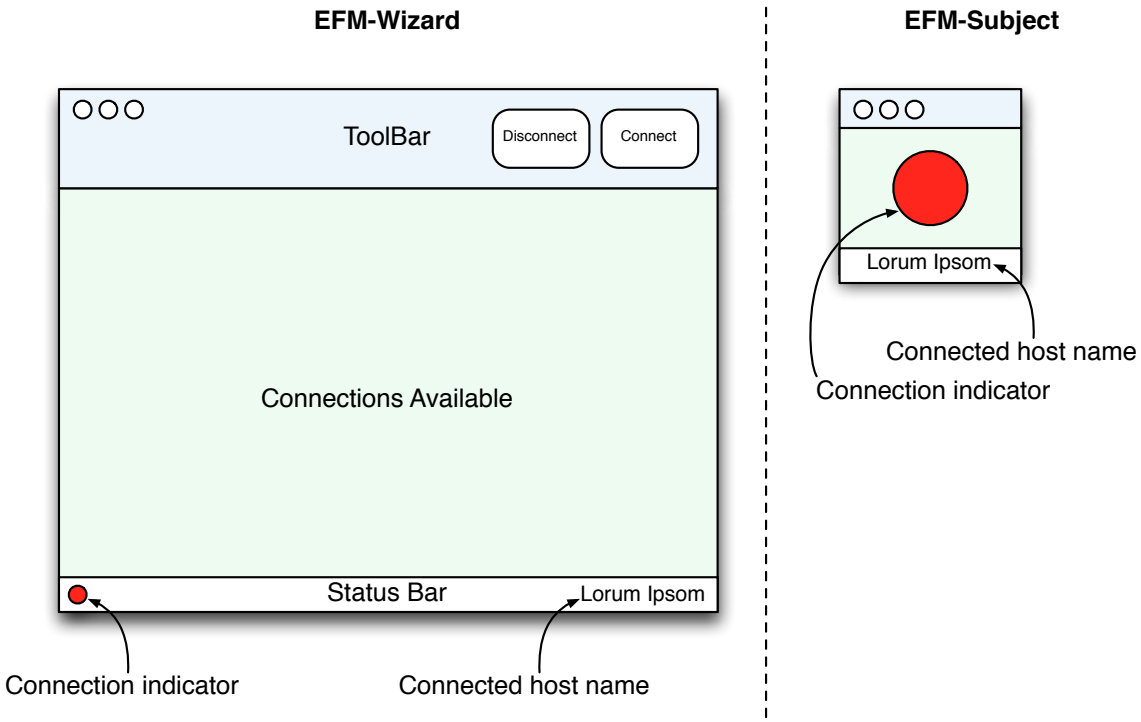

Figure 5-15 - EFM GUI - [left] Wizard, [right] Subject

As with the Visual Matrix and SID tool, the EFM will be built for the Mac OS X platform. Figure 5-16 and Figure 5-17 show an overview of the EFM-Wizard structure and Nib objects. The structure of EFM-Wizard and EFM-Subject are fundamentally the same except there is no message logging functionality in EFM-Subject. As shown in Figure 5-16 the EFM .framework implements the core classes involved with sending and receiving messages. These have been bundled into a .framework so it can easily been included into any plugin development bundle, thereby presenting the available classes and methods to the IDE.

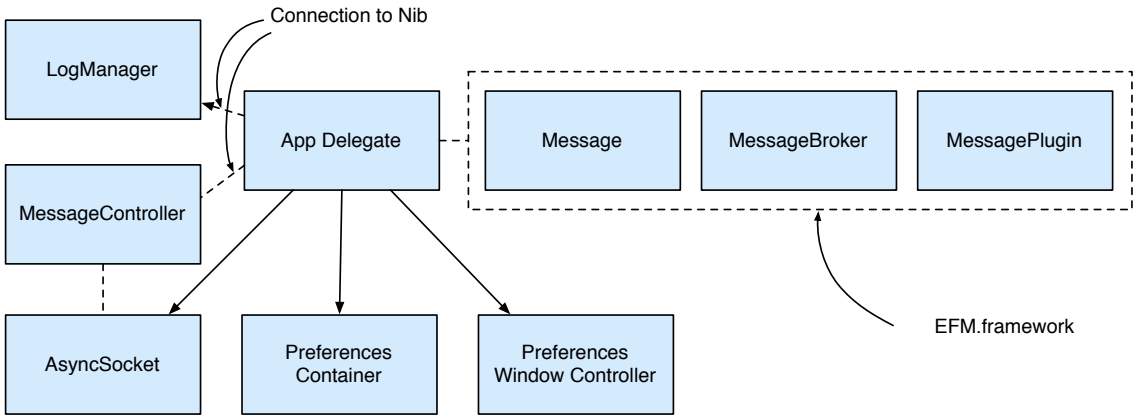

Figure 5-16 - Overview of EFM-Wizard Structure

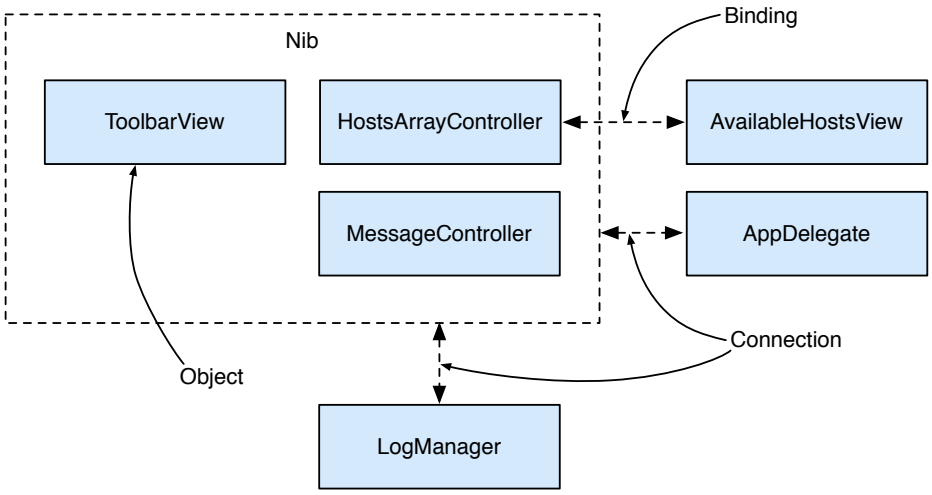

Figure 5-17 - EFM Nib Objects, Example Bindings and Connections

Plugins will be handled within the AppDelegate class. Upon application initialisation, a specific folder will be scanned and any plugin files found that conform to the EFM architecture will be loaded into the application and its GUI presented on screen. Messages will be routed through the host application where they will be passed down to all loaded plugins. Developing plugins for the EFM requires the implementation of a number of methods, which are defined in the .framework as a protocol. These can be seen in Figure 5-18. Using *fire* method as an example, this is used when the plugin is loaded enabling the plugin to perform any actions that need to be completed after loading, e.g. displaying the plugin GUI on screen. A complete overview of the EFM .framework including a guide on how to make plugins is discussed in Appendix A.

```
- (id)initWithType:(int)tType;
- (NSString *)name;
- (void)fire:(id)sender;
- (void)messageBroker:(MTMessageBroker *)broker didReceiveMessage:(MTMessage
*)message;
- (BOOL)respondsToMessagesFrom:(NSString *)pluginName;
```

Figure 5-18 - EFM .framework Required Methods

Messages will be passed over a network using a network socket. The initialisation, and handshaking of this socket, is largely handled by the AsyncSocket class. This class is open-source, developed by the Cocoa community, and it wraps lower-level CFSocket and CFStream classes into a higher-level *fire-and-forget* functionality.

Figure 5-19 shows the base EMF-Wizard graphical user interface. Instances of available EFM-Subject connections are shown in the available connections view. Selecting a host then clicking ‘Connect’ will attempt a connection. Once connected, the connection indicator icon in the bottom left corner will turn green and the connected host name will appear. The subject and the wizard are then able to communicate via the *bolted* on custom GUI plugins. A log is either automatically generated, or if auto logging is disabled from the preferences, a panel is shown asking for a file name and location for the log.

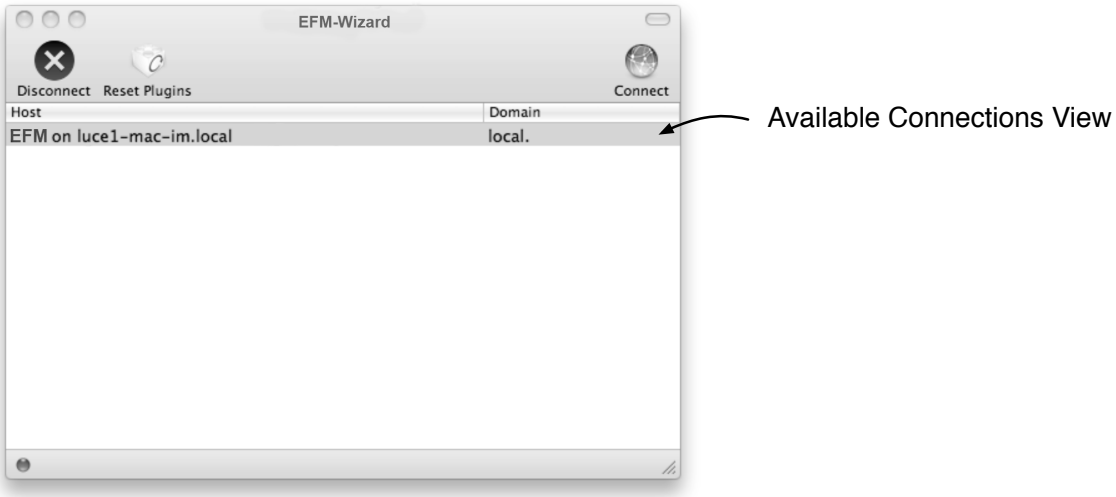

Figure 5-19 - EFM-Wizard GUI

5.3 Summary

This chapter outlined the design and implementation of the SID and EFM framework components. Using the SID tool and the library of SID macro patches, provides researchers with a previously unavailable collection of simulated smart devices that can be easily incorporated into their research. Using these patches, and the resulting simulations, empirical evaluations that require user interaction can be facilitated. The control mechanism used to manipulate multiple simulations in parallel is unique to the SID tool. Using this, wizards have the ability to instantly respond to a subject's interaction in real-time.

The EFM tool is a minor framework component designed to allow both two-way communication between the subject and wizard and to provide researchers with a means to produce desktop GUI interfaces backed with a messaging protocol. As outlined in section 2.6.3, a Wizard of Oz experiment is designed to fool a subject into believing they are interacting with a computer. This two-way communication is not therefore about chatting between a human subject and human wizard, but rather, it has been designed specially for communication between a human subject and what the subject believes is a computer (but is in fact a human). This enables the RME to be more versatile, giving researchers the additional ability to conduct a wider variety of research.

Chapter 6

Prototyping Interfaces

The previous two chapters have discussed framework components for observation, capture and simulation. This chapter looks at the final component of the framework and introduces iProto. This component will be used to produce prototype smartphone interfaces for evaluation in the RME, thus creating a complete framework for designing, prototyping and evaluating smart home control interfaces. Figure 6-1 shows the final iProto toolchain in use. Using the iProto Desktop Authoring application, prototypes can be produced using high fidelity graphical assets. These can then be deployed onto an actual smartphone for evaluation in the RME.

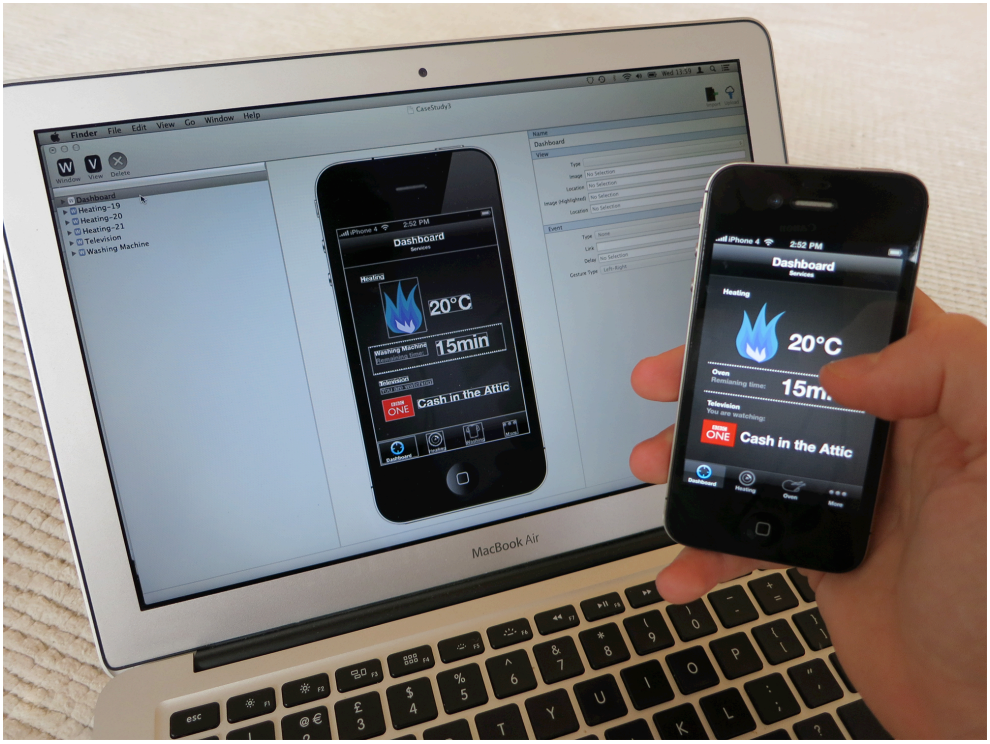

Figure 6-1 - iProto Authoring and Deployment onto an iPhone

6.1 Introduction

Chapters 4 and 5 mainly discuss the issues involved with evaluating interfaces within the context of a smart home environment. Although interface evaluation is essential for delivering incremental improvements it only part of the overall iterative design cycle outline in section 2.5.1. In order to provide a complete solution for facilitating the improvement of user interfaces destined for smart home control, both the evaluation and design need to be considered.

As described by Nielsen [99] [100] the iterative design process of prototyping and UXD evaluation offers real benefits to the usability of user interfaces, both in terms of removing interaction bugs and reconceptualising the interface. Based on the review outlined in section 2.5, it is this process that will be the focus of this chapter.

6.1.1 Rapid Prototyping of Mobile Touch Screen Devices

The goals of using prototyping techniques to analyse touch screen interfaces are in essence the same as those for any type of interface. This is, to acquire feedback from

real users regarding various aspects of the interface design, which can be used to gain knowledge relating to what users found difficult, or what features could be included or removed to ultimately make the interface superior.

With traditional mouse and keyboard driven desktop computers a number of prototyping techniques are well established. These can range from low fidelity paper prototypes to high fidelity mock-up applications. As discussed in section 2.5.6 this thesis will only be concerned with high fidelity prototyping. When developing high fidelity prototypes for touch screen devices all currently established techniques become obsolete, as they are unable to support the additional functionality touch screens provide. For example a traditional desktop computer cannot accept gesture inputs no matter what prototyping tool is used due to the fundamental limitations of the hardware. Although these limitations could be overcome with the incorporation of a touch screen, the physical size of a desktop computer and the difference in form factor prevents the interface being used as it would in the real world. To increase the fidelity and realism, the only real option is to directly run the prototype on an end device.

6.2 Designing a Rapid Prototyping Solution for Touch-based Mobile Devices

iProto is an implemented tool for producing rapid high fidelity prototypes that can be run on mobile touch screen devices. The prototypes produced by the iProto tool take advantage of the available gesture inputs and can be deployed from an internet-based delivery system. This allows prototypes to be deployed to many end devices in various locations. Furthermore, the tool integrates closely with the current design cycle for asset generation used by designers, thereby reducing their workload and the time between prototype iteration cycles. This integration centres on the current software used to produce graphical assets for an interface.

Although the current asset generation cycle can include provision for prototyping, as shown in Figure 6-2, the available prototyping solutions all fall into an asset generation flow that includes a specific stage for producing prototype graphics. Blueprint Viewer (discussed in section 2.5.6), for example, requires that the prototyping interface be

generated within the prototyping application itself; therefore, any end graphics must be redrawn at an increased fidelity within another drawing application. This time consuming process does not lend itself to the idea of rapid high fidelity prototyping because the prototype interface is of a lower quality than that of the final. Furthermore, the inevitable design tweaks, required for an iterative design processes, can happen at both the prototype and mock-up graphic stages.

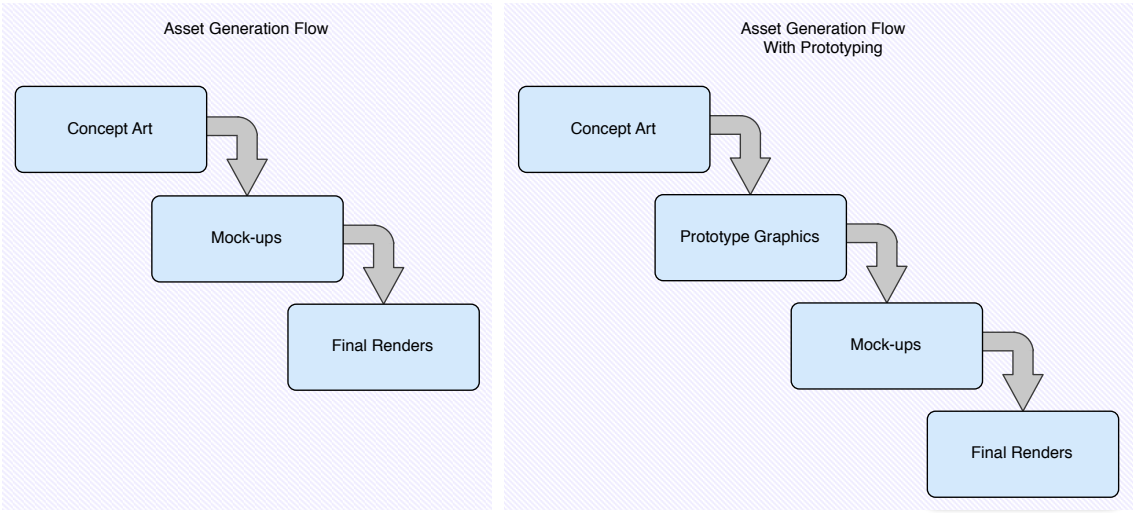

Figure 6-2 - Asset Generation Flow with/without Prototyping

In order to eliminate this, both prototype and mock-up graphics will be rolled into the same stage. In this way the graphics used for prototyping will be the same as those used for the end application, thereby removing any fidelity issues between the end application and prototype.

Used in conjunction with the Reconfigurable Multimedia Environment, the iProto tool is a unique platform for interface prototyping and evaluation. It will be a novel toolchain allowing designers and prototype developers to rapidly produce and deploy intuitive high fidelity prototypes.

Figure 6-3 shows a high-level overview of the iProto. The premise is that graphical assets can be exported from currently used graphic applications then authored into a

functioning prototype. This prototype would then be deployed via an internet-based mechanism onto an end device, e.g. iPhone.

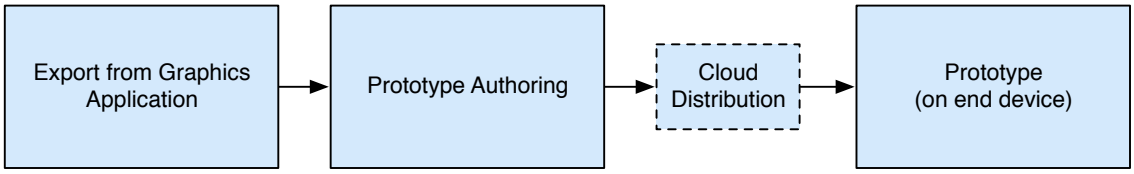

Figure 6-3 - High-level Overview of iProto Tool

6.2.1 Integrating into the Existing Application Design Model

The majority of interface assets will consist of custom designed, hand drawn graphics (e.g. logos, buttons and splash screens) produced by a designer. Although off the shelf graphic elements may be available, these are usually only used to provide consistency amongst different apps. Furthermore, using graphics that are off the shelf can often be ineffective to use due to low resolutions and lack of alpha channel. Figure 6-4 shows a real world example of a button graphic element that was custom drawn to fit into the interface theme. Although the SDK developer (Apple) provided a button graphic, the size and colour gradient were impossible to change.

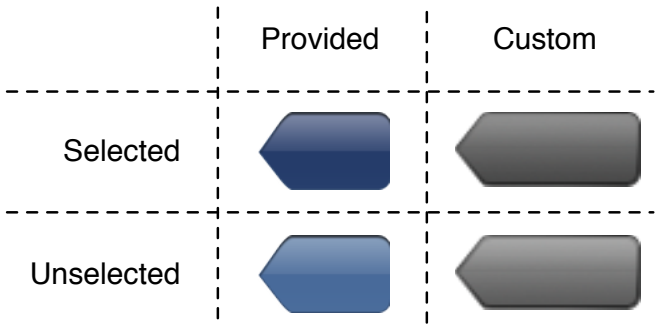

Figure 6-4 - Real World Example of Provided [162] and Custom Graphic Elements

If we consider the design methodology from section 3.1.1, integration with a graphics application, e.g. Photoshop, simply means the efficient movement of data from one

software application to another. The software application in question is a prototyping tool, which takes graphical assets and enables a prototype to be produced and deployed to an end device. One way of achieving this is to use a manual process whereby assets are exported from one application and imported into another application. However, due to the number of individual assets that make up a user interface, a manual method is simply unrealistic. An automated process is therefore required that will export all the graphic assets and relevant information from Photoshop. These need to be exported in an intelligent format so they can be read by another application that will be used to produce a high fidelity prototype. Simply automating a save process is not sufficient as additional information (e.g. image name and location) is needed. Furthermore, the selection of formats is important, as this will determine the quality of exported the assets.

Table 2 shows the required export information needed from a project to successfully reconstruct the interface elements within another application. Using the information from Table 2 a project can be visually recreated in exactly the same form as it was originally created. Because a simple save procedure does not include the required information, an alternative custom export procedure is required.

Object	Information Required
Canvas	size, height
Group	name, z space
Graphic	name, z space, x position, y position

Table 2 - Required Export Information

With the use of plugins, Photoshop can execute scripts for automating processes within the application. Using this type of control it is possible to export all the relevant prototype interface graphics plus the information required for reconstruction. Figure 6-5 shows an example of this export flow.

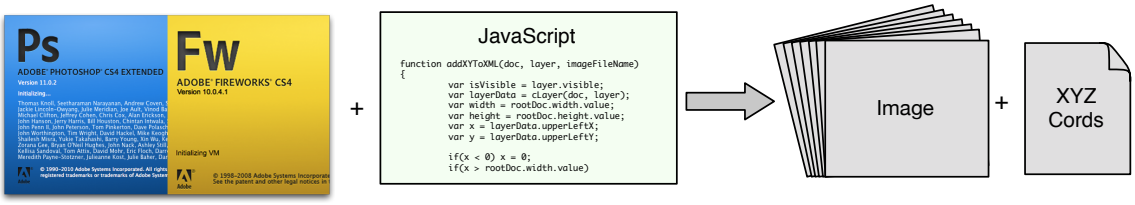

Figure 6-5 - Example Export Flow

Due to the need for transparency, graphics are exported using the Portable Network Graphic (PNG) image format and the additional information, described in Table 2, is stored in a separate file as XML. Pointers to the exported images are stored within the file thereby enabling additional information, such as name or x,y position, to be stored and later matched to the correct image (this can be seen in Figure 6-6).

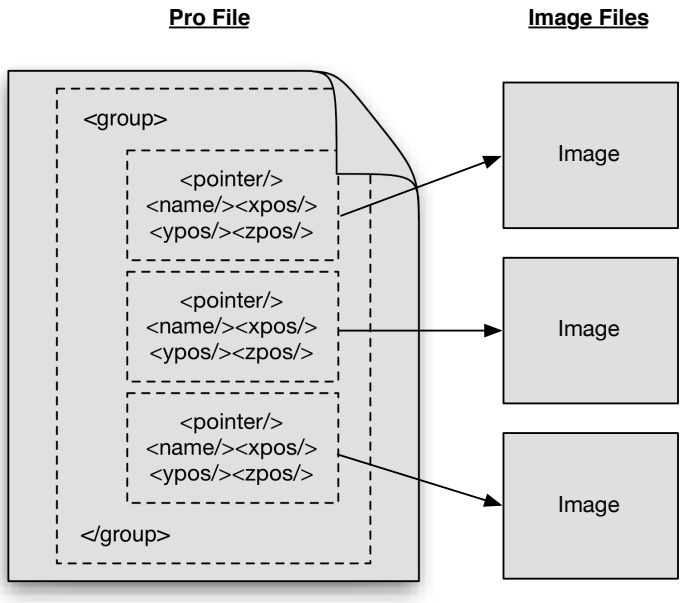

Figure 6-6 - Pro File XML Example Structure With Image Pointers

6.2.2 Rapid Design Cycle

Once all the graphics are exported they need to be authored into a working prototype. If we consider the design methodology outlined in Figure 3-5 this would fall under the second stage. The technique used to produce prototypes is similar to existing techniques outlined in section 2.5.6. This involves linking assets together to create a pseudo-realistic application generated solely from high fidelity images. Figure 6-7 shows a real world example of how this type of linking works. Asset one is an image-backed *window* that holds a number of image-backed *views*. The *views* are linked to other *windows*. Tapping on a linked *view* loads another *window* (and corresponding *views*). Figure 6-7 has six linked assets *views* (2 to 7) each of which load a different *window* (shown on the right).

Figure 6-7 - Asset Linking Example

Although similar, the iProto system does in fact build upon the existing techniques in three crucial ways. First, the iProto tool does support all standard types of gesture control; a default tap, a timer or a gesture can trigger the activation of an asset's links. When linking assets together a user can change the trigger option for that specific asset. Second, it facilitates rapid production of high fidelity interactive prototypes from pixel-perfect graphical assets. And finally, it will allow for the direct import of assets from applications such as Photoshop and the rapid linking of those assets to produce a prototype.

Figure 6-8 shows the practicalities of this technique whereby a user simply drags one object onto another object within a hierarchical view. By default this link will be activated when a user taps on the asset; however, this interaction can be changed using the additional information panel. Using this technique a user can quickly create all the necessary links thereby producing a high fidelity pseudo-realistic prototype.

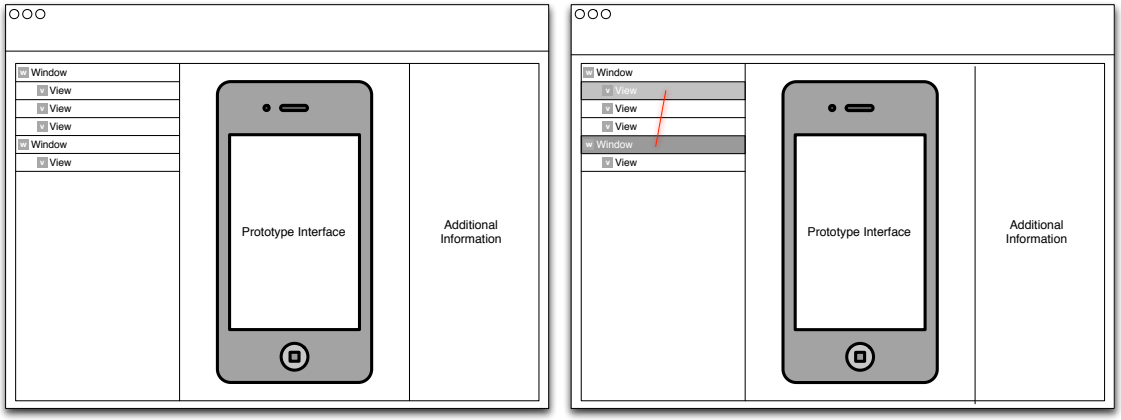

Figure 6-8 - Rapid Asset Linking Example

When importing assets into iProto the same layout and order defined within Photoshop will automatically be transferred. A group in Photoshop will be treated as a new *window* and a layer as a *view*. For example using the information stored within the additional XML file, an asset drawn in Photoshop at position 20, 176 (x,y) with a size of 114px,

111px (width, height) will then be drawn in the same place and displayed with the same size within the iProto system. An example of this can be seen in Figure 6-9.

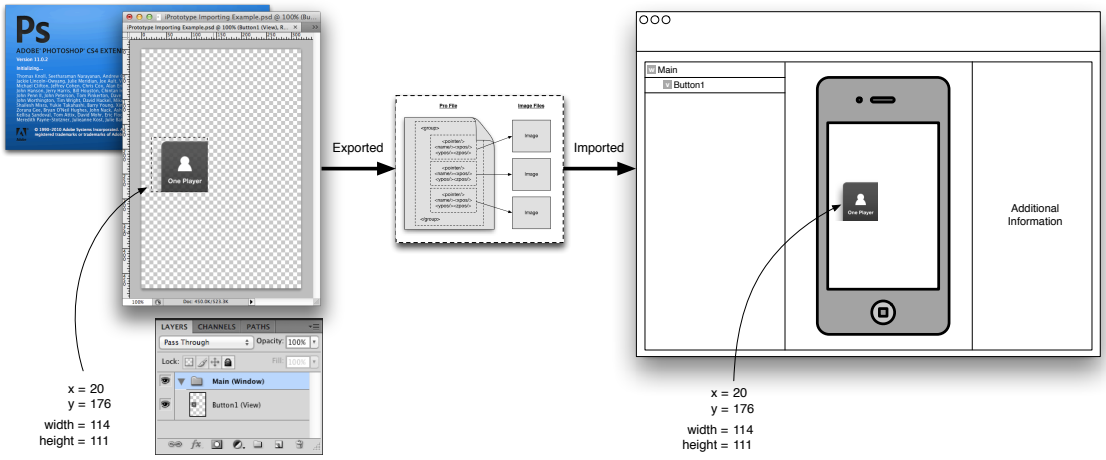

Figure 6-9 - iProto Desktop Publisher Importing Example

The iProto system will also support the layer order defined by Photoshop. Using the preserved layer order, the prototype interface will take on the same appearance (for example, buttons will continue to be placed on top of the background image). As with any view-based interface, when these assets are translated into views the furthest forward in the stack will be the first to respond to user input.

One crucial element to the iProto tool is the ability to quickly make adjustments to prototypes. Following the design methodology described in section 3.1.1 these adjustments would be made in the desktop graphic application (e.g. Photoshop) so all design stages are working from a common set of assets. Adjustments made in the desktop graphics application will be reflected within the iProto system after the assets are re-imported. When re-importing assets a smart search will automatically re-assign the current linking information based on assets and links already included within the project. Figure 6-10 shows an example of this automatic re-link procedure using two assets that have been connected together by a user. If these assets were re-imported without using the automated link procedure, then any information regarding the link, including custom settings, would be lost. Using the automatic re-linking enables the

links and associated information to be kept after the underlying graphic assets have been changed.

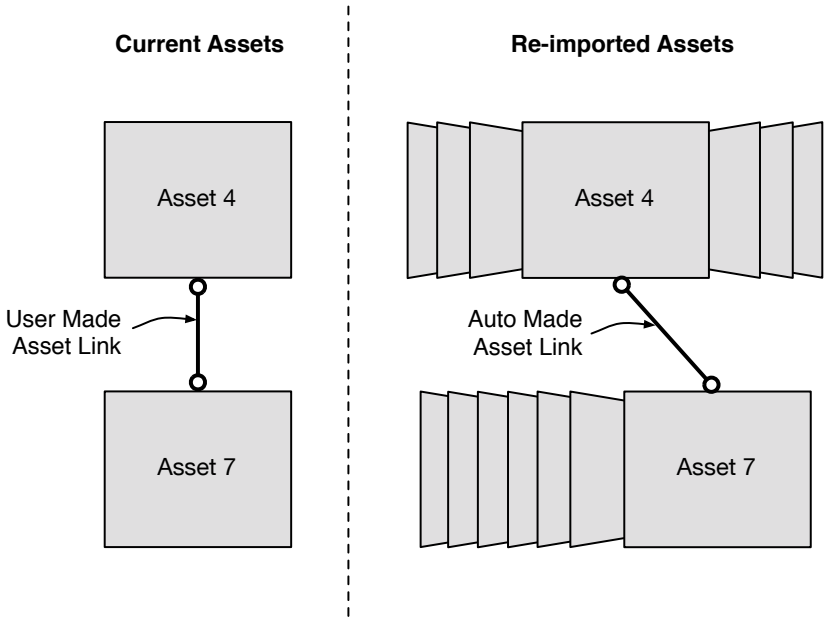

Figure 6-10 - Automatic Linking of Re-imported Assets

6.2.3 Rapid Distribution

For the rapid distribution of a prototype, to an arbitrary number of devices located in various places, the best option is a cloud-based distribution mechanism. With this devices need only be connected to the internet to have access to all prototype files stored within the cloud. Figure 6-11 shows the basic principle of the prototype distribution mechanism. The iProto desktop publisher first packages up assets and related information into a single project file. It then uploads this to a file server and adds a record of its existence into a database.

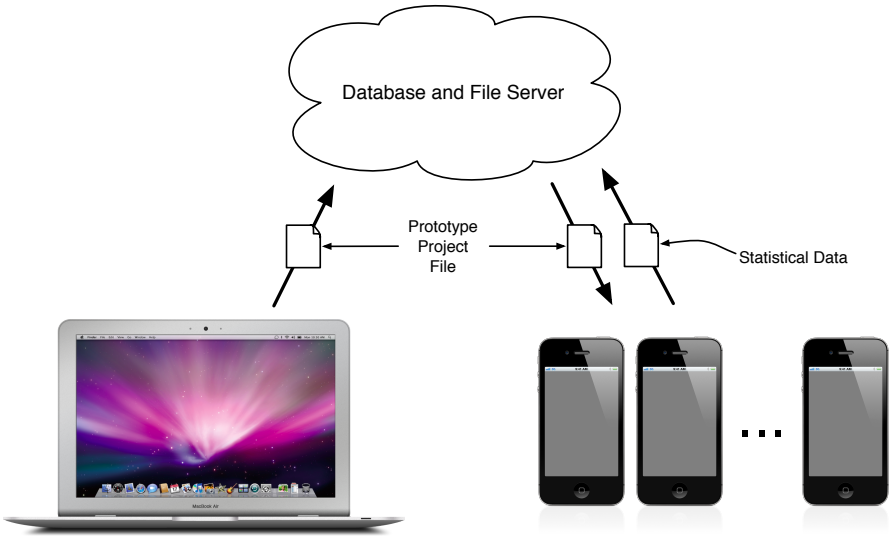

Figure 6-11 - High-level Overview of Cloud-based Distribution Mechanism

6.2.4 Rapid Deployment to a Mobile Device

To enable prototypes to run on an end device the iProto tool includes a dedicated mobile app. This pulls prototypes from the cloud and runs them on a device.

Traditionally an app regularly indicates to a user its presence, or current function, by displaying visible updates. To produce a prototype with an immersive experience any indication of this underlying app is an unwanted distraction and it should therefore be relegated to an invisible background process. A service menu is still needed, however, to select the prototype in use or to change specific settings. Using the advantages provided by a touch screen interface, a number of successive taps (four in particular) will give access to this service menu (as shown in Figure 6-12). This type of user interaction is almost never needed within an app so it is unlikely to interfere with the running prototype. When in the service menu a specific prototype can be selected that will then download from the server.

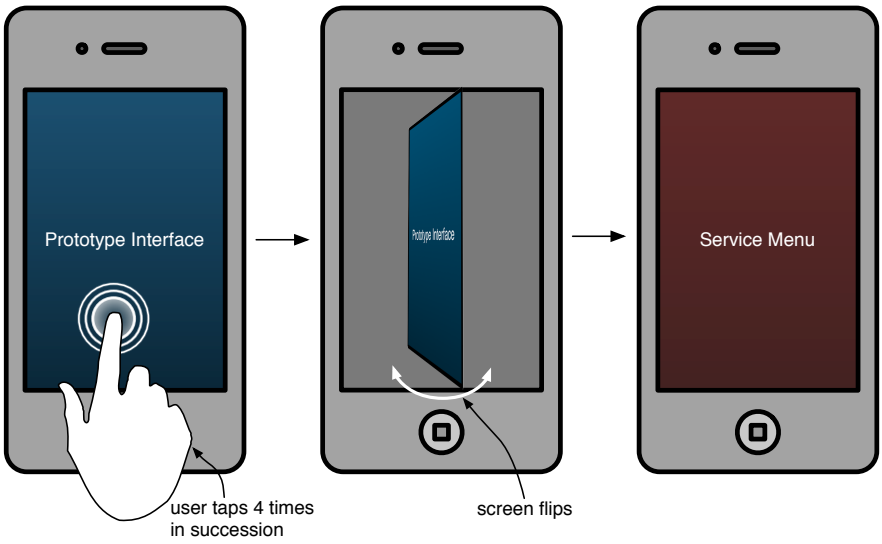

Figure 6-12 - iProto Mobile App - Service Menu Access

6.3 Implementation

The implementation of the iProto tool consists of four main areas (as shown in Figure 6-3): first, the development of the export mechanism for exporting assets from a graphics application. Second, the development of the iProto Authoring application. Third, the development of the cloud-based distribution mechanism, and finally the development of the iProto Mobile App (used to run prototypes on an end device). These four main areas are closely interlinked causing an amount of blurring between the development of these four separate components.

Based on previous experience, and considering the available hardware, the iProto tool will be developed for the Apple Mac OSX and iOS platforms. Using the Mac platform not only allows for tight integration with the RME and SID components, but also takes advantage of the tight integration Apple has developed between its hardware components and software layers. For example, the same application programming interface, Cocoa, can be used on both Apple's desktop and mobile platforms.

One major benefit of Cocoa for the development of both the desktop and mobile applications is the ability to use Core Data. Core Data (previously discussed in section 4.3), is a framework for automating common tasks associated with an object's life

cycle. Core Data uses an automated external data store to hold all data saved from the managed object context. When a project is saved this external data store can be accessed as a regular file. All data relating to the project is therefore stored in a single convenient location that can be copied quickly into the cloud.

If the Managed Object Model for both the desktop and mobile versions of Core Data are the same then data can be read and written by either the desktop version of Core Data or the mobile version. Moreover, the iOS implementation of core data has been heavily optimized for mobile devices to ensure a high performance.

Figure 6-13 shows an overview of the technologies that are used within the iProto framework component. These have been selected based on experience, availability and ease of integration.

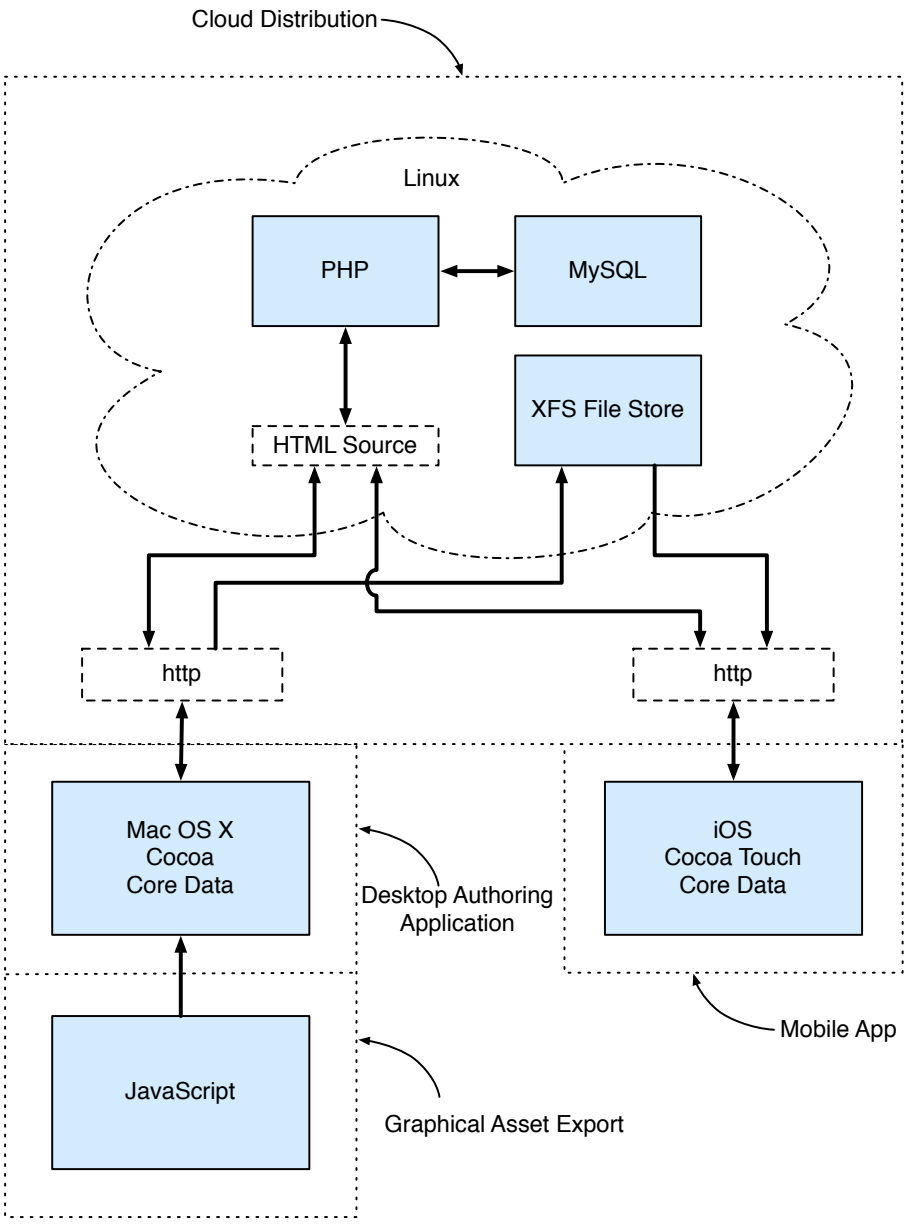

Figure 6-13 - Overview of Technologies used within the iProto System

6.4 Photoshop Plugin

The Photoshop plugin script exports all graphical assets for iProto. The script performs two major roles: the exporting of individual layers as image assets and the exporting of related data as an XML file.

The Photoshop community has previously covered exporting individual layers and scripts are already freely available, one script in particular is supplied from Adobe as example code and it is this script that is used as the basis for the Photoshop plugin. It must be noted that the original script is the work of Naoki Hada and Tom Ruark [167] and although it is not directly the work of the author its functionality is described below to enable the reader to distinguish between the original base code and the additional work completed by the author.

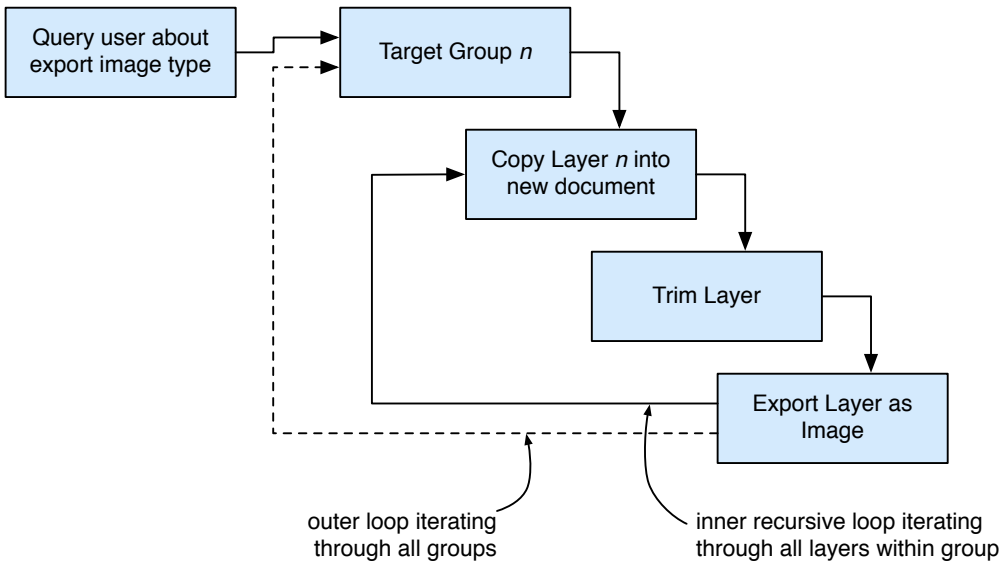

Figure 6-14 - Original Layer Export Flow [167]

Figure 6-14 shows the original layer export flow. The essence of the script is two loops. The outer loops through all groups within the project. The inner loops and exports all the layers within a group. If we compare Figure 6-14 and Figure 6-15 we can see a significant amount of modification is needed to export the additional layer and group information that is required for iProto.

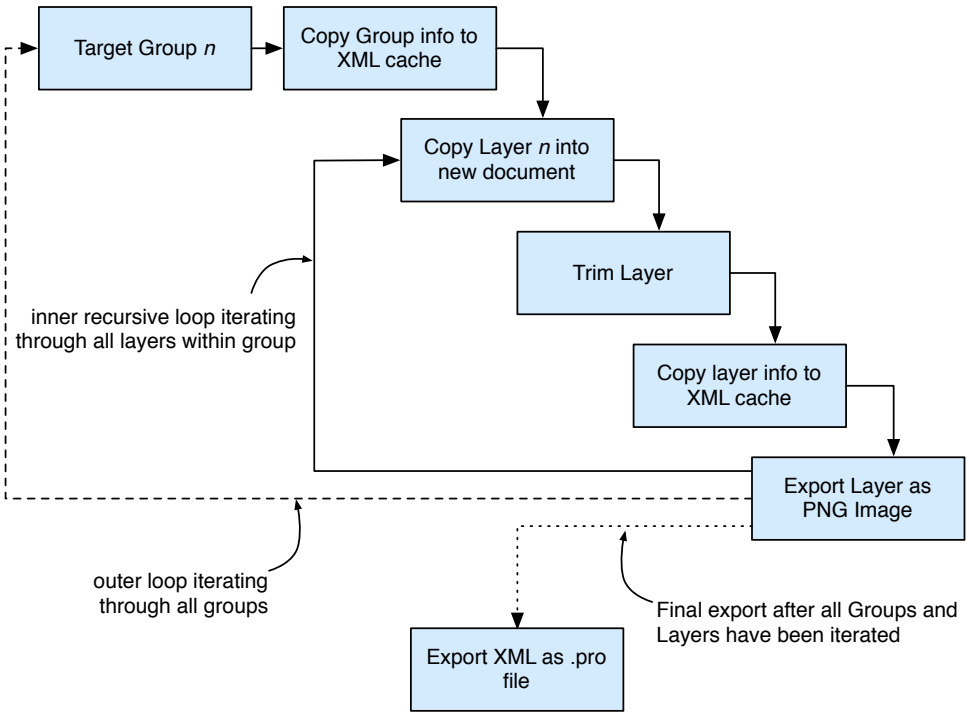

Figure 6-15 - Modified iProto Photoshop Export Script Flow

Upon initialisation an XML cache is generated with the necessary header data. Whenever the group loop completes a full iteration, the new group information is copied to the XML cache. Whenever the layer loop completes a full iteration, the layer information is also copied to the XML cache. When all information has been added to the XML cache the footer data is added and a final *.pro* file is exported to a user specified location.

A naming system is also added to enable exported asset names to be unique and human readable, should individual asset files need to be tracked. Below is the format used for asset naming. An additional benefit of using this format is the ability to reconstruct a projects assets into the correct groups and layer order even if the accompanying .pro file has become corrupted or lost (lack of coordinate data would prevent the assets from being placed at the correct point in the coordinate system).

<GroupNumber>_<GroupName>_<LayerNumber>_<LayerName>.png

6.5 Desktop Authoring

The iProto desktop OS X application will support the authoring of prototypes. The application will handle the importing of assets, connection of the assets and the uploading of the final prototype into the cloud. The four main development areas are; Core Data support, Import Manager, UI and Upload Manager.

The iProto tool uses a similar design methodology to author prototypes as that of other general view-based applications. Within this section any discussion of a view that is preceded by a p (e.g. pView or PView) is referencing a view that holds a graphic asset and is destined for an actual deployed prototype. Other views discussed are simply ones that have been produced for the desktop authoring application.

6.5.1 Core Data Support

Core Data is used to automatically manage data using a Managed Object Context (MOC) and Persistent Store Coordinator. Changes to the data stored within the MOC are either user driven, using the desktop UI, or automatically driven by the import manager. Cocoa Bindings is used extensively to connect data (in our case the MOC) to the front end GUI using a controller to mediate the two.

Figure 6-16 shows the Core Data Model Schema used within the iProto system. The model can be split into two distinct elements, visual data and connection data. Most of the data held by the Core Data Model (CDM) is specifically for the generation and authoring of prototypes; however, some data (e.g. isLeaf Boolean) is used for the UI of the desktop application (for example the isLeaf Boolean is used by the navigation outline view tree structure).

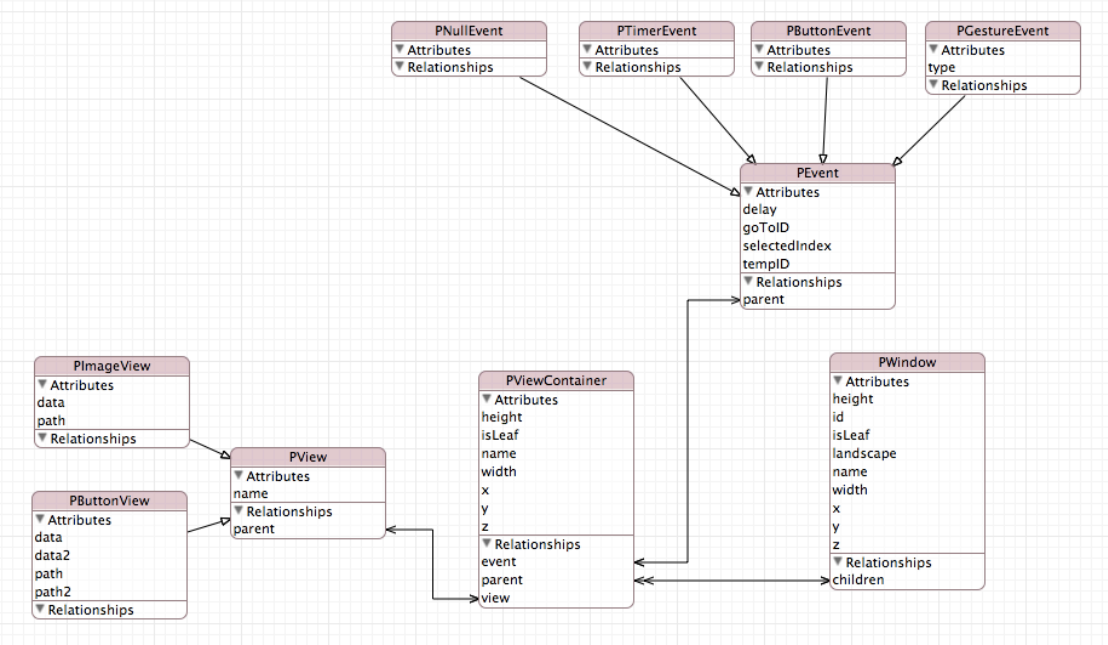

Figure 6-16 - Core Data Model

Figure 6-17 shows the visual information held by the CDM. Below is a description of each entity and the data it stores.

- *PWindow* - Holds base data for the prototype, for example, the height and width attributes define the screen size of the device. The id attribute is unique for each *PWindow* and is used to create links between *PViewContainers* and *PWindows*. A *PWindow* has one-many relationship with *PViewContainer*.
- *PViewContainer* - Holds generic *PView* data common to all *PViews*. The width, height, x, y, z attributes are used to define and location of a *PView* within a *PWindow*. A *PViewContainer* has a one-one relationship with *PView*.
- *PView* - A parent entity.
- *PImageView* - A *PView* that will display an image. The data attribute will hold binary image data and the path attribute will hold the location of the image for reference.
- *PButtonView* - A PView that will display a button. It will hold two images (on image and off image). The data and data2 attributes will hold binary image data, path and path2 will hold the location of the image for reference.

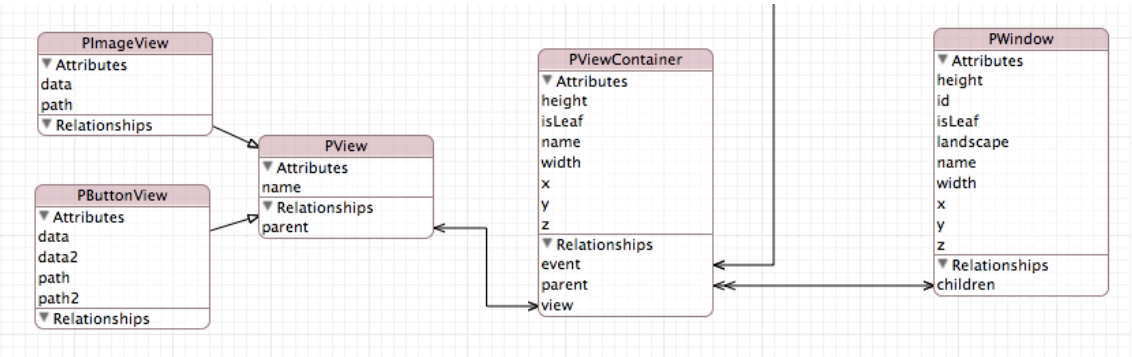

Figure 6-17 - Visual Prototype Data Held By The CDM

Figure 6-18 shows the connection information held by the CDM. Below is a description of each entity and the data it stores.

- *PEvent* - The parent entity, which holds data common to all PEvent entities. goToID holds the unique id defined within PWindow. delay holds a delay time between the event being activated and the link being executed.
- *PGestureEvent* - Holds additional gesture type data and used to respond to gesture input.
- *PButtonEvent* - Holds no additional data and is used to respond to a tap input.
- *PTimerEvent* - Holds no additional data and is used to setup a timed event.
- *PNullEvent* - Holds no additional data. Used as an initial holding event, which responds to no input.

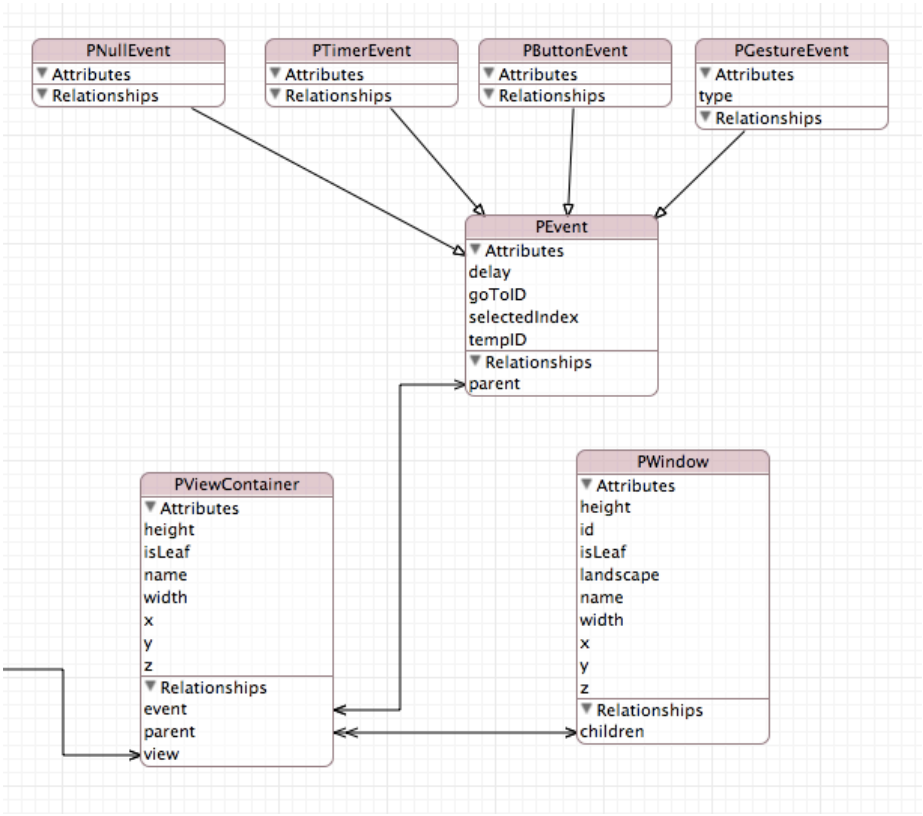

Figure 6-18 - Connection Prototype Data Held by the CDM

Figure 6-19 shows the block diagram for the iProto Desktop Authoring application.

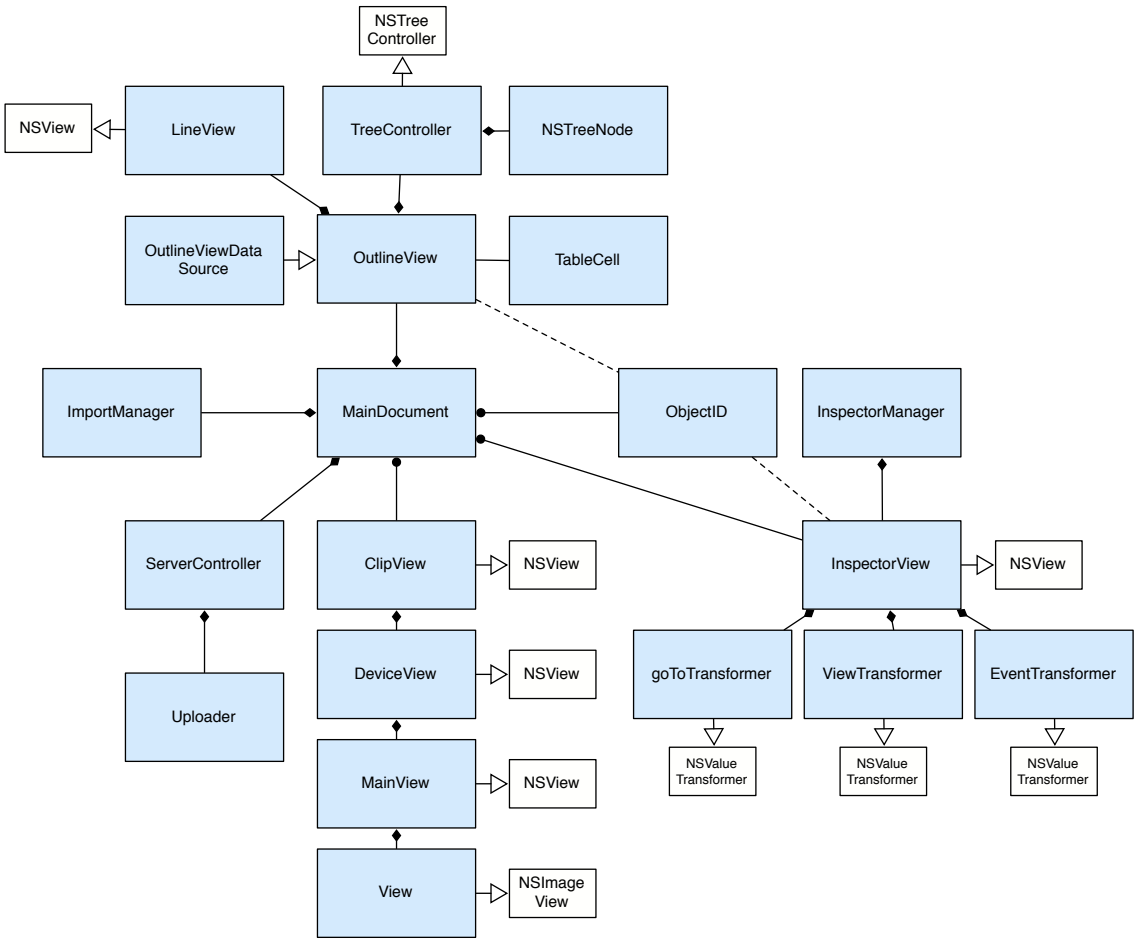

Figure 6-19 - iProto Desktop Author Block Diagram

6.5.2 Import Manager

Figure 6-20 shows the flow sequence implemented to import assets into the iProto Authoring application. It also includes the steps for implementing the smart search and the automatic linking of existing assets within a project. The *Import Manager* class interacts directly with the Managed Object Context; adding, deleting and modifying the data, which is then reflected via a number of controllers in the GUI. The class is not explicitly linked to the rest of the application and can easily be disconnected or not used. This enables a user to import assets and configure a prototype manually should they feel the need.

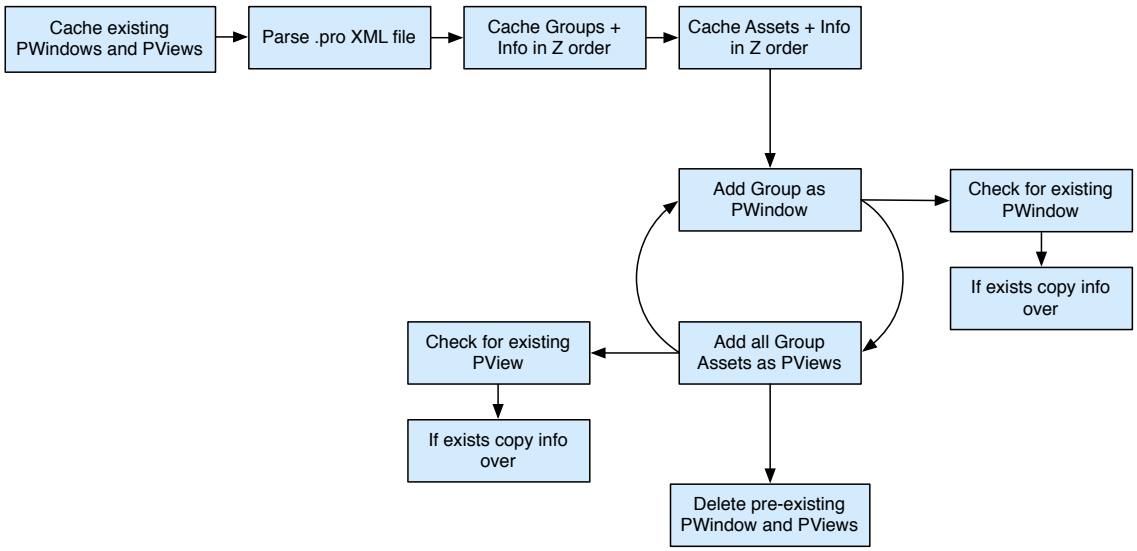

Figure 6-20 - Import Manager Flow

6.5.3 User Interface

As with a lot of applications, the user interface takes a lot of development time and requires a large amount of code. To keep track of the various UI components the UI is split into four main areas. First, the Outline and Connections View. Second, the Main View. Third, the Inspector View, and finally the Toolbar (shown in Figure 6-21).

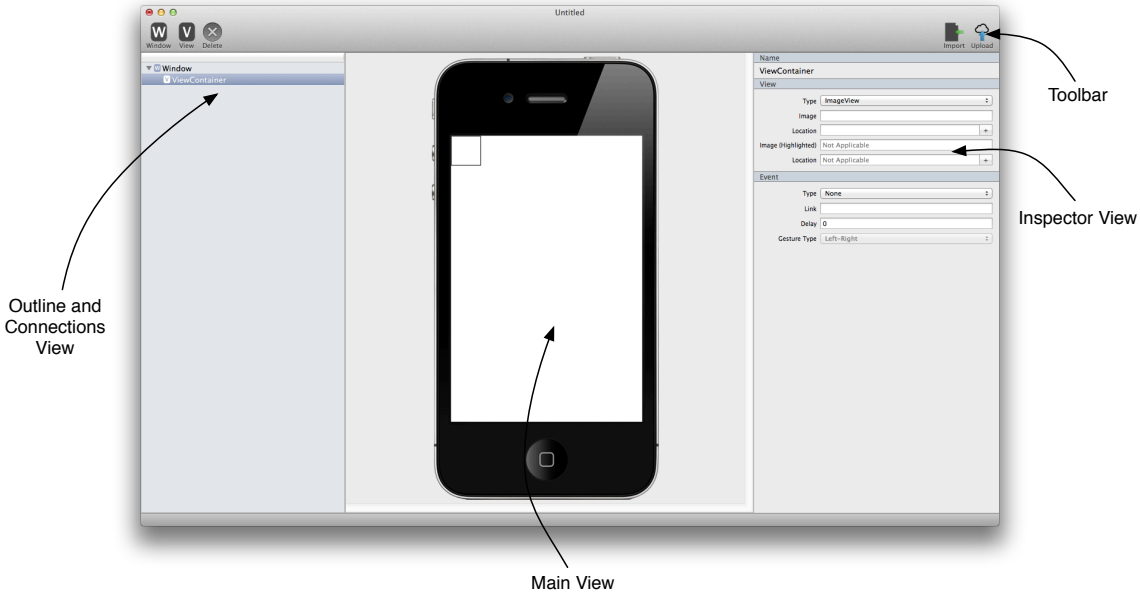

Figure 6-21 - iProto Desktop Authoring Application UI Layout Overview

The initial GUI development used Interface Builder (discussed in section 4.3) for the basic visual layout and structure. Figure 6-22 shows the interface objects instantiated by the nib, including all major GUI elements. Amongst the objects instantiated are PView and PEvent. These are not traditional objects; rather they are *managed* objects bound to the selection of the TreeController object. Whenever a user changes the selection of the outline view, this selection is automatically reflected by the TreeController.Selection reference. Binding this reference to a managed object enables an object to be automatically updated with different data, while still being presented to other objects as a single instance. This is especially useful for extracting GUI data, based on a different user selected variable (in this case the selected TreeController object).

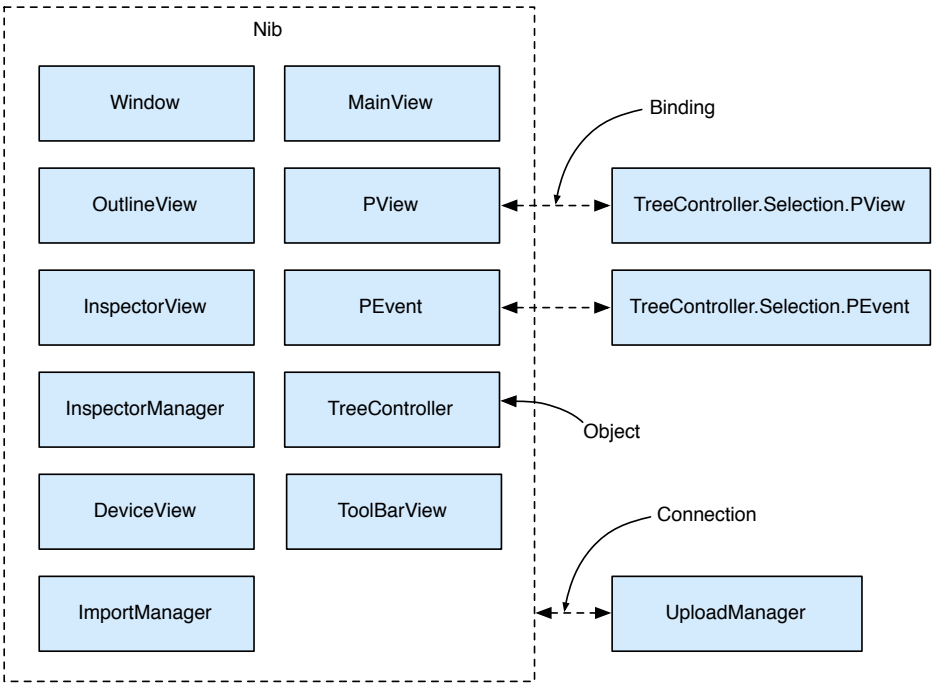

Figure 6-22 - iProto Desktop Application Nib Objects, Example Bindings and Connections

Within the iProto desktop application the bindings are used extensively to link data from managed objects to GUI elements. Figure 6-23 shows how the PView object is bound to the selection of the TreeController object. This enables the *path* string variable to be automatically displayed using a NSTextField. Also shown in Figure 6-23 is the

use of a value transformer, this enables the transformation of data into a different, more useful format. In the example shown, the transformer removes the bulk of the *path* string, leaving just the filename.

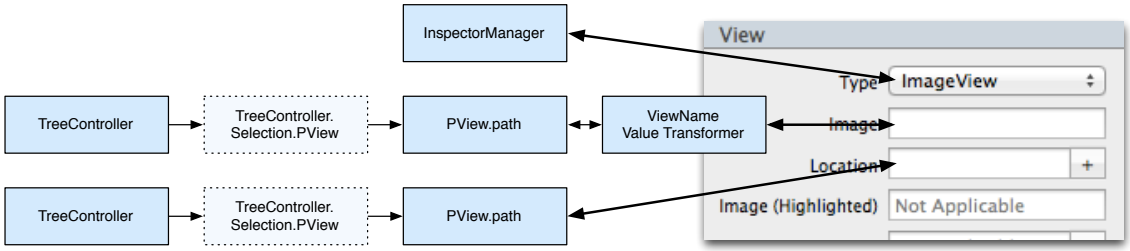

Figure 6-23 - InspectorView GUI Bindings Example

Although the use of bindings speeds up development, it can only be useful for basic data display and manipulation. The InspectorManager object controls the more complex parts of the InspectorView (e.g. changing the PView type).

The MainView is used as a preview window, enabling the user to have an idea of how the final prototype will look on a device. The blue overlay displayed in Figure 6-24 shows the different PViews currently being rendered for preview. A user can move any of the PViews around within the MainView by simply clicking and dragging it around. A PView will also automatically expand or contract to the size of it's underlying asset.

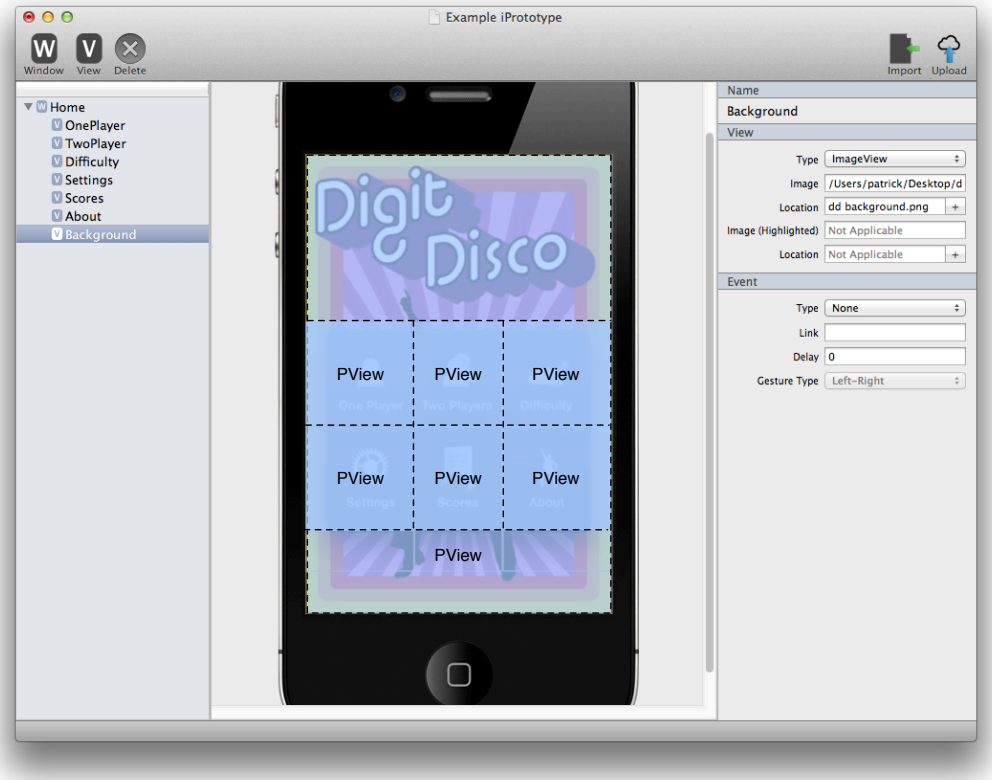

Figure 6-24 - MainView Displaying PViews GUI Example

The final major GUI component is the OutlineView. This is used to select PViews or PWindows and to create links between them.

Figure 6-25 shows how a user would connect a PView and PWindow together by first selecting the PView and then dragging a connection to a PWindow (connection mode is entered when the Ctrl key is held down). The connection *Event* information is then automatically populated in the InspectorView using a default ButtonEvent and the name of PWindow, which will then be displayed.

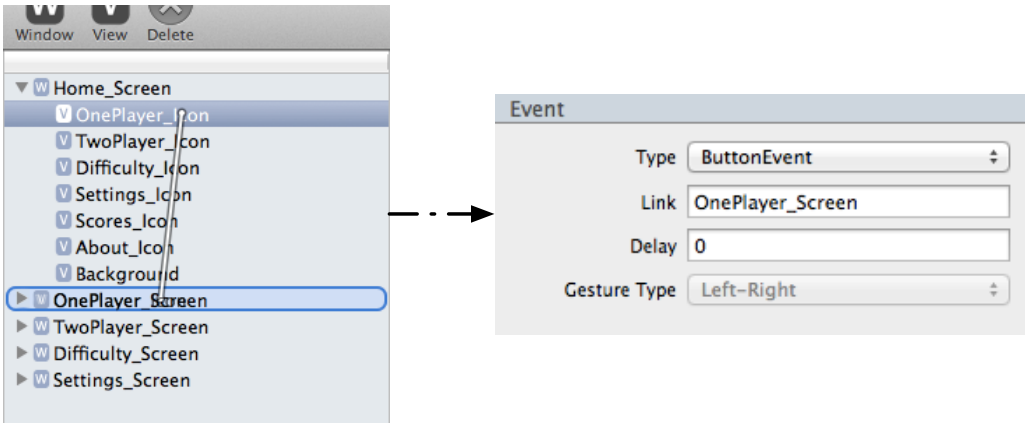

Figure 6-25 - Making a Connection Between PView and PWindow Example

The OutlineView can also be used to re-order PViews so that different PViews can be placed behind others. When importing assets this ordering process is completed automatically by the ImportManager, reflecting the same layer order as that in Photoshop.

6.5.4 Upload Manager

The task of uploading files to the cloud is managed by two classes: *ServerController* and *FileUploader*.

ServerController handles communication with the MySQL database while the *FileUploader* class handles the movement of files from the host computer to the cloud based file store. Shown in Figure 6-26 *ServerController* submits queries to the MySQL database using HTML fronted PHP scripts. The requests are sent over the internet using the standard HTTP protocol. Included within this class is an XML parser; this enables information to be sent back from the webserver, for example, a list of available prototypes currently being stored for a particular username.

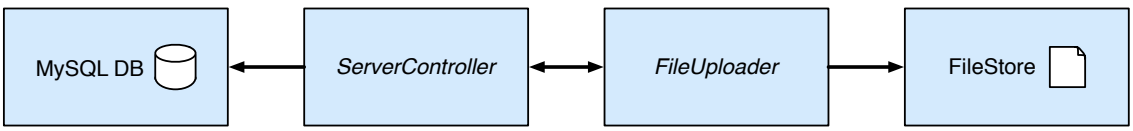

Figure 6-26 - Server Communication Configuration

The *FileUploader* class is created from standard open-source code for transferring files using an HTTP POST method. The class transforms the data into an URLRequest and *POSTs* the file onto the file store.

6.6 Cloud Distribution

The cloud distribution mechanism provides the glue between the desktop authoring application and the mobile prototype tool. It follows the overview shown in Figure 6-27.

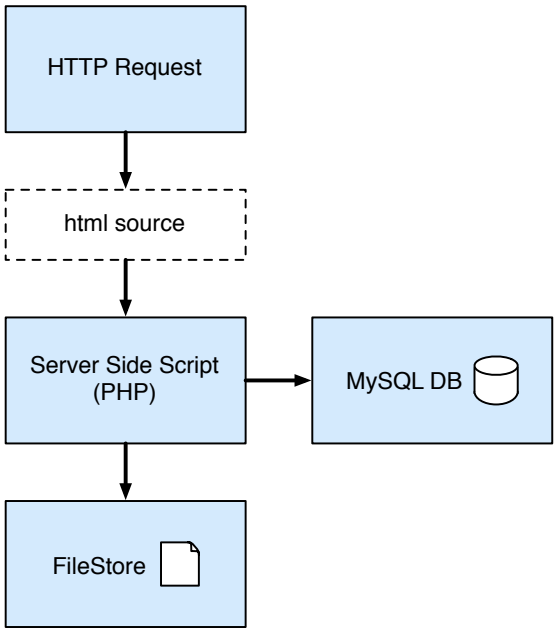

Figure 6-27 - Overview of Cloud-based Distribution Mechanism

A MySQL database holding information on users and files is accessed via PHP with embedded MySQL queries. PHP is a stateless scripting language, meaning whenever the code is initialised no previous unsaved information can be drawn upon within the script. Whenever any process is requested, variables are passed to the PHP script within the URL, then extracted using the PHP *GET* method.

Figure 6-28 shows the server-side code structure used within the cloud distribution mechanism. The available functionality currently includes server status check, insertion and deletion of files, file information requests and database formatting. All external

controllers interface with the database using the DatabaseController class, via the SecurityController class thereby preventing MySQL injection hacks.

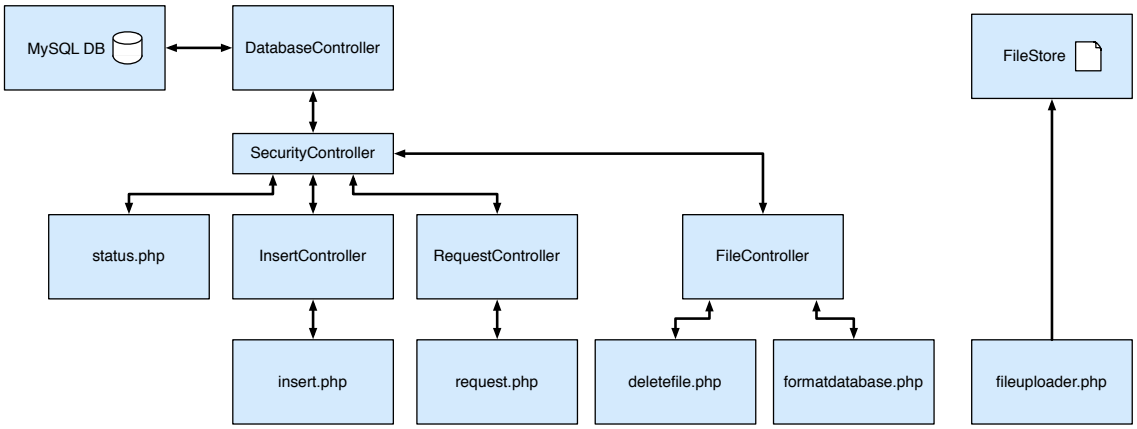

Figure 6-28 - iProto Server Side Code Structure

Figure 6-29 shows the detailed process of uploading a project file to the cloud. The username and password of the user is stored within the desktop publishing application enabling the automatic upload of project files to the database. This *one click* upload and *one click* download solution enables a rapid distribution mechanism for mobile-based prototypes developed using the iProto system.

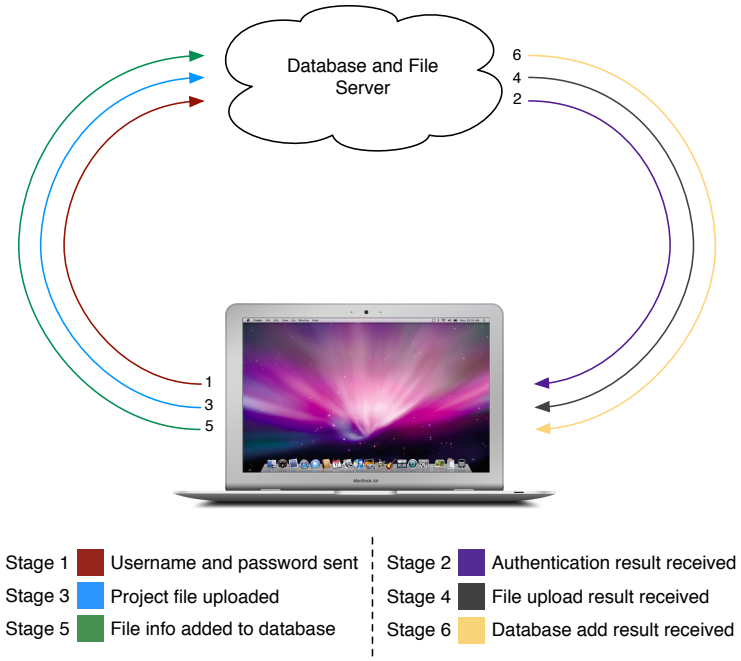

Figure 6-29 - Project File Upload Process

6.7 iPhone App

The deployment device for a prototype produced with the iProto Authoring Application is an Apple iPhone smartphone. The iProto iPhone App has been developed for the iOS mobile operating system to execute the prototype on the device. The apps main role is to communicate with the cloud distribution mechanism, download and unpack a prototype file (outputted from the iProto Desktop Authoring application) and manage the prototype while in use by the subject. Additional functionality includes an interaction overlay and the management, storage and upload of statistical data regarding prototype usage.

Due to the need for the apps interface to be nearly invisible, the work for the app will mainly consist of background processes such as communicating with the cloud, setting up prototypes and executing links (either via taps or gestures).

When launched, the iProto mobile app will query the database for a list of available prototypes and, upon user selection, will download the project file, unpack the

prototype and execute it. After the interface evaluation exercise has been completed statistical data can be uploaded from the smartphone back to the cloud thereby creating a global store of all data.

The iPhone App structure can be split into two key areas, frontend (e.g. UI) and backend (e.g. server and file management). Figure 6-30 shows the basic app structure split into these two key areas. As with many apps the AppDelegate interfaces between the frontend, backend and, because the app is based on Core Data, the persistent store. The persistent store is a file located on the local disk that holds all the data used by Core Data. Because this data is held as a single instance it can easily be replaced with a new or modified file.

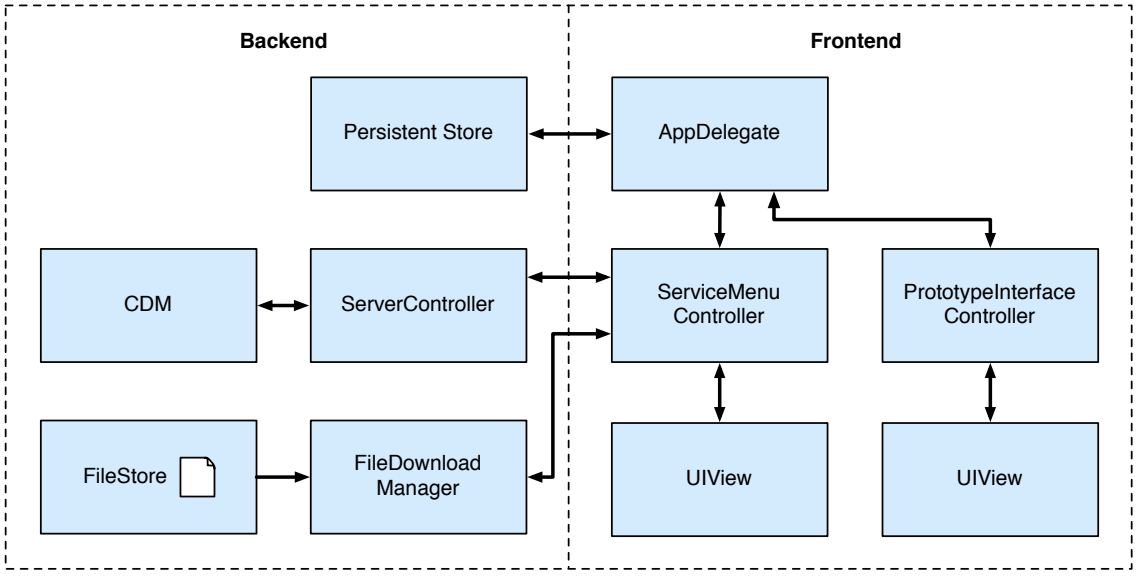

Figure 6-30 - iProto iPhone App Structure

The iProto iPhone App persistent store is based on the same Core Data Model (CDM) as the iProto Authoring application and is used to hold data relating to the currently downloaded prototype. The prototype source file, downloaded from the cloud distribution mechanism, contains all the data needed to display the prototype on screen. Because the CDM is the same across both mobile and desktop platforms (as shown in Figure 6-31) the downloaded prototype source file can be used to simply replace the

existing data held in the persistent store without any additional processing. The Managed Object Context will still be capable of reading the data because the CDM is exactly the same for both old and new data sets. This makes downloading and updating prototype data between the iProto Authoring Application, the cloud and the iProto iPhone App very efficient.

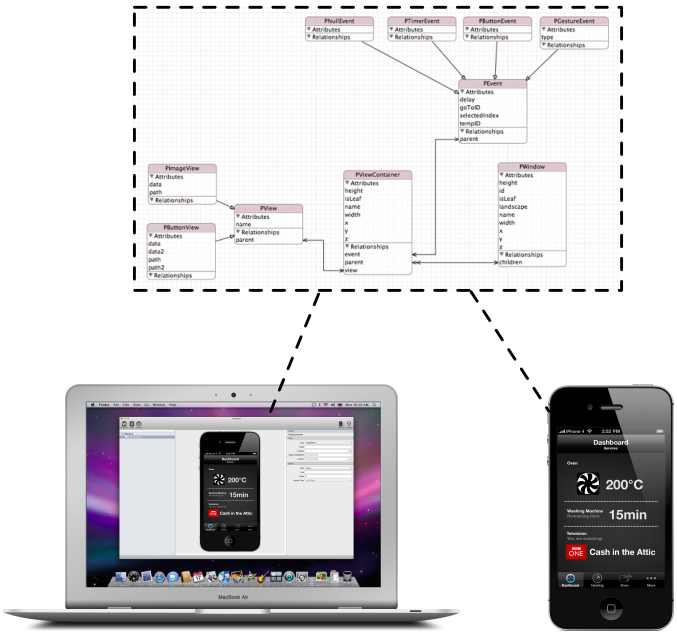

Figure 6-31 - Use of Same Core Data Model with Desktop Authoring App and iPhone

Figure 6-32 shows the persistent store replacement flow for loading a new prototype downloaded from the cloud.

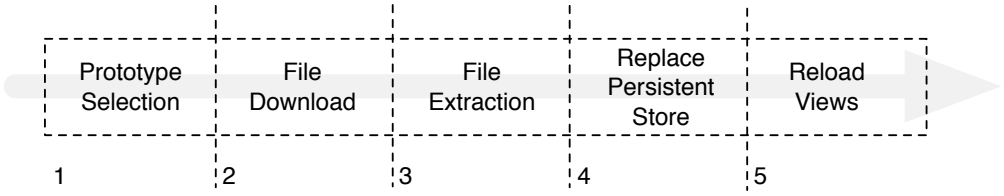

Figure 6-32 - iProto iPhone Persistent Store Replacement Flow

Regardless of what data is being held in the persistent store it is used by the PrototypeInterfaceController as the source for rendering a prototype onto the device's screen. Figure 6-33 shows the flow sequence implemented to display PViews onto the iPhone screen. Once the PViews are sorted into the correct z-order the PrototypeInterfaceController class translates them into standard image-backed iPhone UIViews and using the stored x and y coordinates, the view is display on the screen. Events are also attached to the views, which are triggered when the user interacts with the display (the event type is dependent on the type specified during the prototype design).

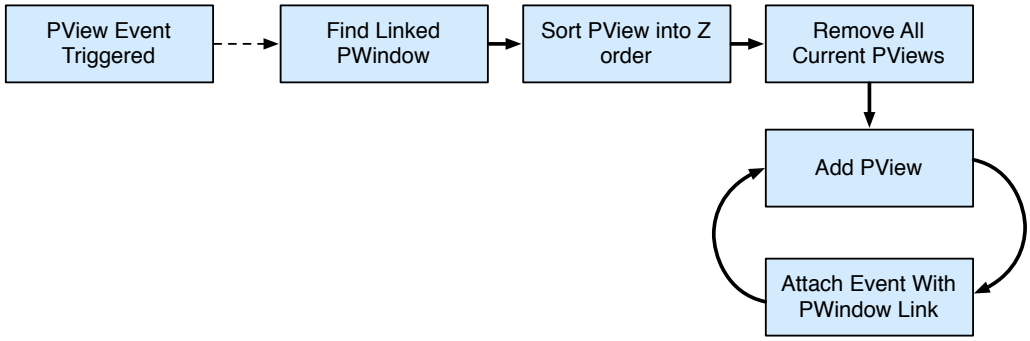

Figure 6-33 - iPhone PrototypeInterfaceController Adding PViews Flow

The PViews are first extracted from their PWindow parent object then sorted using an array sorter. The sorted array is then iterated over and the relevant data (e.g. x, y, width, height) is stored as local variables for later use.

The ServiceMenuController class manages communication between the app and the cloud distribution mechanism. ServerController is used to query the database on available prototype files and a list is displayed to the user. Once a user selects a file the FileDownloadManager class handles the download request and caching of the file.

The interaction overlay is used to show the wizard (or analyst) how the user interacts with a prototype by overlaying the gestures they perform (this is discussed further in section 4.2.1). It is managed by a top-level class, which registers all UITouch events being received by the device. These are intercepted at the highest-level of the app. An

image is drawn on the screen where the subject's finger is placed using the provided coordinate information (shown in Figure 6-34). The event is then passed down the app event chain for use within the rest of the app. An example of an event overlay is shown in Figure 4-8.

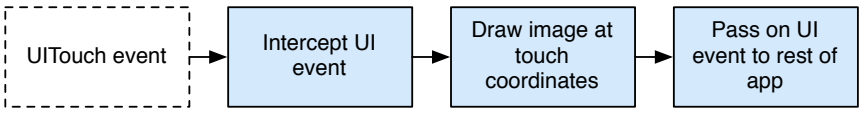

Figure 6-34 - UITouch Event Interception

6.8 Summary

This chapter has introduced iProto as a toolchain for creating and distributing high fidelity interface prototypes for smartphones. An example of a final prototype running on an end device can be seen in Figure 6-1.

The original aim of the iProto system was to facilitate the iterative prototype process for smartphone interfaces. This has been achieved through a re-thought design methodology including the development of a number of bespoke software applications. The use of the toolchain facilitates the production of smartphone user interface prototypes for smart home control.

Upon conception, the iProto system was unique both in its design, and in the functionality it provided for the development of smartphone interface prototypes. Since its development, however, a similar system has been presented by Jørgensen et al. [168] detailing a similar process to the iProto system. Details of this research are discussed in section 8.2.2. With the high pace of interest in smartphone research, it is perhaps unsurprising that two independent research groups were analysing the same problem and although the two systems are similar there are sufficient differences to classify both as being novel. Specifically the system outlined by Jørgensen et al. is based mainly on web technologies whereas the iProto system is built entirely from native technologies (this is discussed further in section 8.2.2).

Chapter 7

Case Study

This chapter evaluates the framework through a case study. The individual component parts of the framework are tested and evaluated from the point of view of a user experience designer, along with the framework as a whole. A notable addition to the framework is discussed at the end of the chapter, where a SID-based feedback mechanism is integrated into iProto following limitations discovered during the case study.

This case study assesses whether the framework can be used as a round-tripping toolchain for facilitating the prototyping and evaluation of smart home control interfaces. It demonstrates the successful use of the framework in providing an efficient process for facilitating the design, prototyping and evaluation of such interfaces. This thesis therefore reinforces the prototyping and evaluation process, allowing for increased research and insight into the user experience of smart home control.

The three main components of the framework are the focus of this case study. The chapter will first discuss the design and development of interface prototypes, and subsequent round-tripping of those prototypes with both major and minor modifications. Second, it will discuss the design and development of two SIDs that relate to the previously defined prototypes. Finally the RME will be discussed, in particular its use by an Open University research team to conduct an empirical evaluation.

Figure 7-1 shows the workflow when using the framework to prototype and evaluate smartphone interfaces for smart home control. Evaluating the framework from the point of view of a user experience designer is appropriate as they are the intended user of the RME, iProto and SID tools.

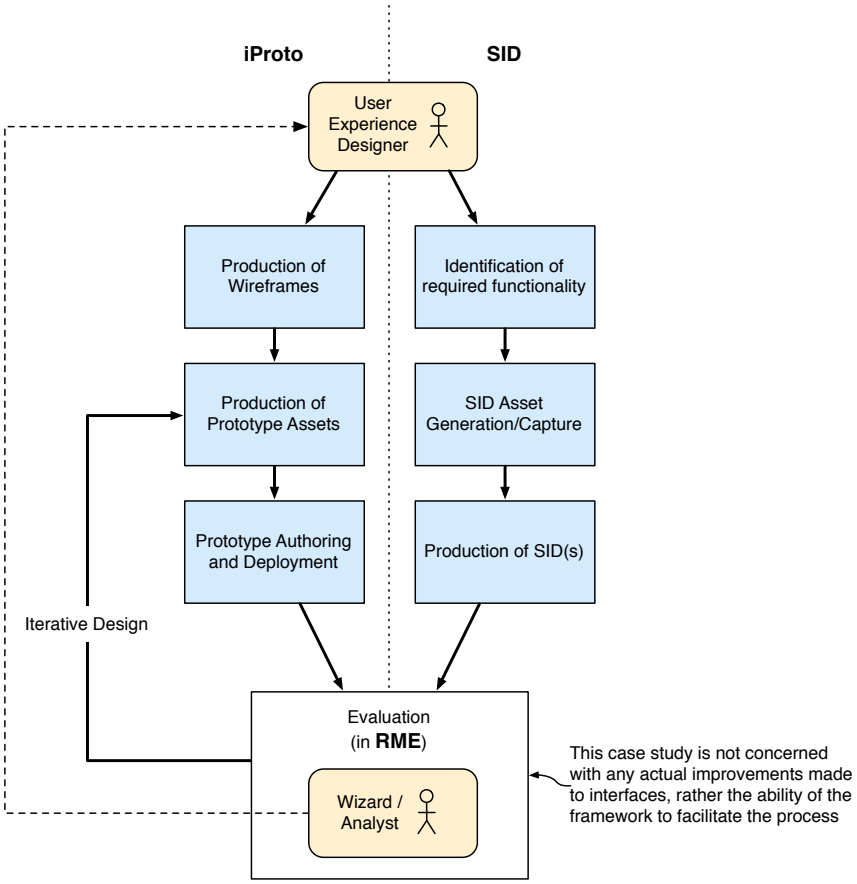

Figure 7-1 - Workflow when Using the Framework to Prototype and Evaluate Smartphone Interfaces for Smart Home Control

7.1 Conceptualising the Prototypes

The prototype will focus on two commonly controlled devices within the home; a heating controller and a washing machine. These were selected due to their ubiquitous nature within homes and because the type of control they invoke enables the framework to be fully utilised and assessed. Both the heating controller and washing machine also require a number of core controls to be implemented to provide functionality.

Using the iProto toolchain a number of prototype interfaces will initially be wireframed, then mocked-up in Adobe Photoshop, authored into working prototypes and deployed onto a smartphone. Two SIDs will also be produced as representations of a heating controller and a washing machine. These will provide the necessary feedback for the subject during evaluation without the actual need for the physical devices. Alterations will also be made to the prototype, which will be deployed back onto the smartphone enabling the evaluation and refinement process to be repeated.

7.2 Production of Prototype Interfaces

As outlined in section 2.5 the user experience high-level design flow begins with the formation of conceptual ideas, leading into the production of task flows and wireframes. It is at this point that the iProto toolchain latches onto the current process and augments it. With the ability to use graphical assets produced after the wireframes, a working prototype can be authoring that can be deployed onto an end device. Changes can be made after an evaluation exercise, and the prototype can once again be deployed back onto the device. The prototype production process can be seen in Figure 7-2. It is important to note that the process augments the current user experience design process outlined in section 2.5 and that the production of the prototype requires no traditional programming experience, allowing it to be used by less technical users (e.g. designers).

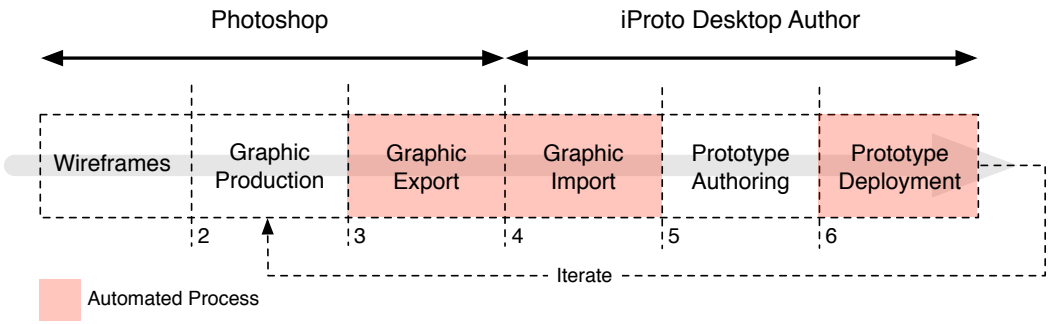

Figure 7-2 - Prototype Production Work Flow Stages 1-6

Looking at the process outlined in Figure 7-2, stages 1-3 cover the development of wireframes and graphics. These stages would normally happen as part of any interface

development and are the stages that are augmented by the iProto toolchain. Stages 4-6 cover the prototype authoring process and the deployment of the prototype onto an end device. Because stages 1-3 are already part of the existing standard interface development pipeline, a user experience designer does not require any additional knowledge for the production of prototype assets. Although stages 4-6 require knowledge of the iProto Desktop Authoring application, the requirement is minimal with the most complex part of the process being the linking of assets, something that is achieved using a common mouse drag and standard inspector panel.

7.2.1 Stage 1

The production of wireframes gives the basic understanding of the layout and information included within the app. Figure 7-3 shows the wireframes developed for this case study. Four major windows will be used, a dashboard, heating controller and washing machine controller and television. With these as a reference, the development of the graphics can begin.

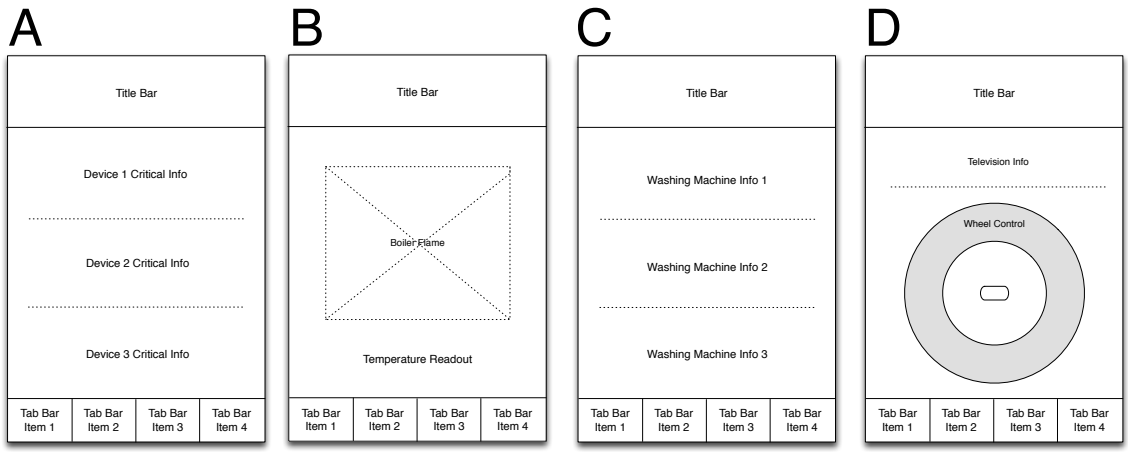

Figure 7-3 - Development of Wireframes, [a] Dashboard, [b] Heating Controller, [c] Washing Controller and [d] Television Controller

7.2.2 Stage 2

With the omission of a separate stage for the production of prototype graphics (as shown in Figure 6-2), the production of pixel-perfect, high fidelity, graphical assets can be completed in a desktop graphics application. Although this case study will focus on the use of Adobe Photoshop, as discussed in section 6.2.1, the iProto toolchain allows for seamless integration with any graphics application that conforms to the iProto export specification (currently this integration is limited to Adobe Photoshop). This provides further integration potential should user experience designers not wish to use Adobe Photoshop. Figure 7-4 shows a detailed overview of the assets used for the creation of the Dashboard window; with each interface component being separated into individual layers. The tree structure used within Photoshop will be transferred to the iProto Desktop Authoring application. The one rule imposed upon the designer is the organisation of top-level folders. Within Photoshop top-level folders are used to represent windows, with all subsequent layers used to represent views. Simply keeping the various components organised allows for swifter authoring of the prototype.

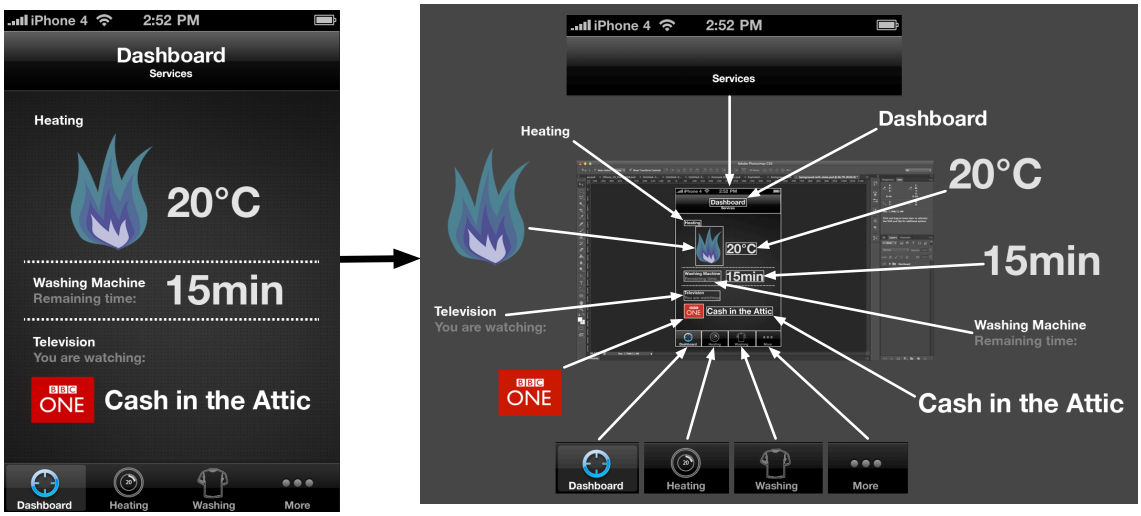

Figure 7-4 - Development of Graphical Assets (Dashboard)

Figure 7-5 shows the final four windows used for the prototype interface. Each window has been organised into an individual top-level folder, which, when imported into

iProto, will be displayed as such. Figure 7-6 shows the addition of two more windows to represent the gestured-based interaction that could be used to increase the temperature of the heating controller. The inclusion of gesture-based control within the iProto toolchain allows real smartphone user experience paradigms to be tested with users, within then context of smart home control.

Figure 7-5 - Individual Prototype Screens

Figure 7-6 - Creation of Additional Screens for Gesture-based Interaction

7.2.3 Stage 3 and 4

Once the graphical assets have been produced, they need to be exported from Photoshop. Exporting the assets is as simple as selecting the *Export for iProto* item from one of Photoshop's menus. After selecting a location, the script exports all the individual layers as separate image files and the related data as XML. The associated information file is also saved. Importing the assets into the iProto Authoring application is achieved by simply selected this information file. This process is demonstrated in Figure 7-7. The export and import of graphics is an automated process that requires no additional user involvement.

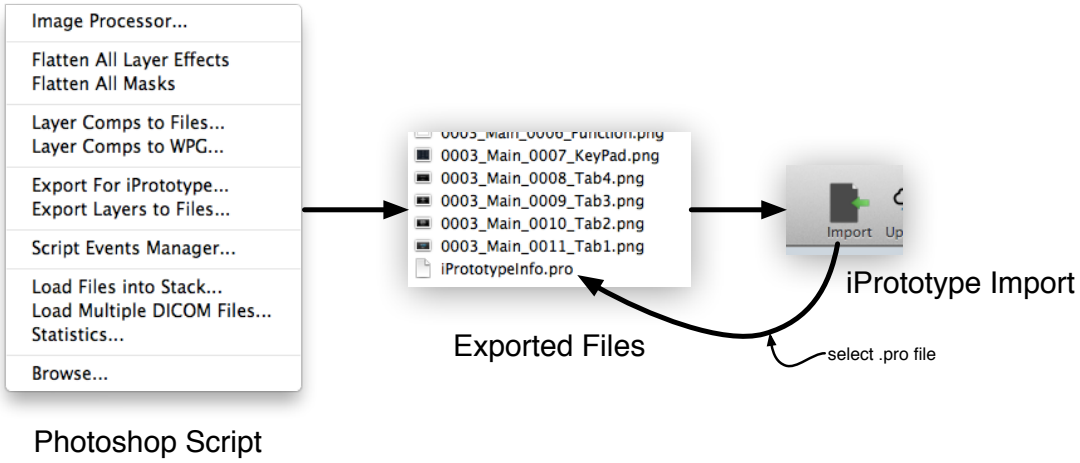

Figure 7-7 - Exporting from Photoshop and Importing into iProto

7.2.4 Stage 5

Once imported, the creation of links between the different assets and the customisation of a links action settings allows the user to determine the desired level of interactivity. As shown in Figure 7-8 (left hand side) the tree structure shows top-level folders as windows and all child layers as views. Views may be linked to windows by selecting the view and dragging and connecting a line to the desired window. As part of this customisation, gestures can be used as a trigger to load different windows, this also shown in Figure 7-8 (right hand side).

Figure 7-8 - Adding Links to Assets

Depending on the complexities of a prototype, authoring it can take a variable amount of time. With the example shown above the creation of 10 links took approximately 40 seconds. The more complex the prototype the longer the initial authoring will take. However, once created, major and minor changes can be made to the prototype without the loss of any links. Additional and modified links can also be made with the inclusion of more windows or major interface changes. This decreases the time required for round-tripping of prototypes thus enabling a faster re-deployment following any modifications.

7.2.5 Stage 6

Once the necessary links have been created the prototype is ready for deployment. Clicking on the *upload* button is all that is required to distribute beyond the local machine. In the background the prototype is packaged up and sent to the iProto cloud servers. Opening the iProto Mobile App and querying the database enables the user to then download and run the prototype on an end device (shown in Figure 7-9).

Figure 7-9 - Prototype running on an end device

7.3 Refining and Round-tripping the Prototype

When refining a prototype all graphic modifications are made using the original desktop graphic application e.g. Photoshop. As outlined in the requirements, section 3.5, the iProto toolchain has been designed to allow both minor adjustments to prototypes and for major interface changes.

Figure 7-10 shows a minor adjustment being made to a prototype. Here, the heating visualisation has been changed in colour (from blue to orange). With a simple

adjustment such as this, rendering out all the assets and re-importing them into the iProto Authoring application may be unnecessary. The individual component can be manually exported and then imported into iProto. The automatic export process can still be used, but exporting an individual asset reduces the time overhead when compared with exporting all other, unchanged assets.

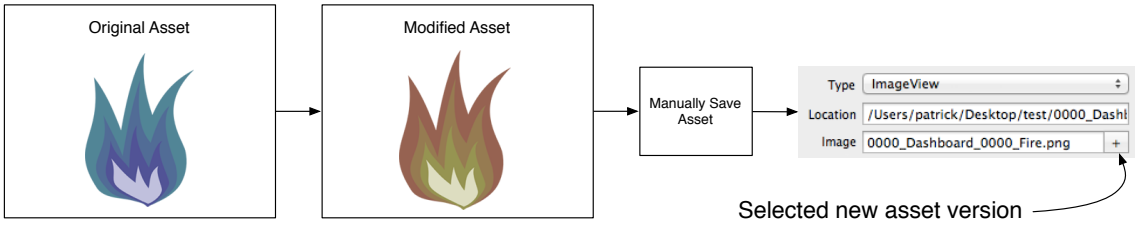

Figure 7-10 - Minor Adjustment to Dashboard Heating Visualisation

Once imported, as all the links between views and windows still remain, the prototype can simply be uploaded to the cloud as described in section 7.2.5 – Stage 6.

Major changes can be implemented in two ways; either by modifying the layers within a particular top-level window folder or by creating entirely new windows. By using new windows, entirely different interface designs to be quickly swapped out for new ones. If a top-level window folder is not linked to, it essentially becomes hidden from the subject. In this way many radically different interface designs can be produced with the links between the specific views and the windows being re-set to point to a different design.

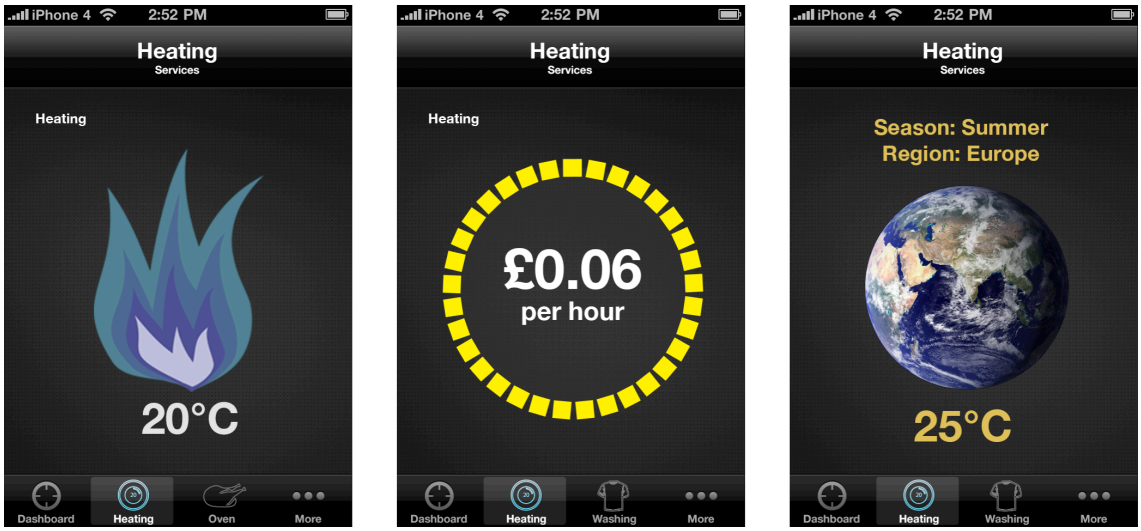

Figure 7-11 - Major Design Changes to Heating Controller

Figure 7-11 shows a major design change to the heating controller. By re-setting the links, the different controller designs can be displayed. It should also be noted that because all these iterative interface adjustments are made in the same application (Adobe Photoshop) when the final design is completed the graphical assets can simply be rendered out and included in the final application. This is an important difference between the tools presented in the current literature and the iProto toolchain. Because iProto fits within the current design workflow, the high fidelity assets used for the prototype are exactly the same as those which will be incorporated into the final app. The ability to use the same, pixel perfect, graphical assets in both the prototyping stage and the final app can therefore been described as a novel contribution to the field of smartphone prototyping research.

This case study of the iProto toolchain shows the process undertaken by a user experience designer in producing a prototype, including the round-tripping of major and minor changes. Having completed the case study one notable improvement has been identified for the iProto Authoring Application. This is the reallocation of links, on mass, from multiple views to an alternative window. This would further facilitate the replacement of windows by automating the change of links from an old window to a new one. However, this improvement does not detract from the current functionality of

the prototyping process. The ability to target one mobile phone over another does not appear to be a problem but more interesting devices, e.g. an iPad may be something to take into account for future work in this area.

7.4 Simulating a Device

The SID tool is a means of providing video-based simulated devices normally found within a domestic environment for interaction between a subject and a prototype interface.

As described in section 5.1, the production of a SID is based on Quartz Composer and the use of a bespoke SID macro patch library. These macro patches determine the type of manipulation applied to an input asset when MIDI input signals are received.

The production of SIDs is both creative and logical. The type of device drives the majority of the production process, from the capture or generation of assets to logical wiring within Quartz Composer. This case study focuses on the generation of two SIDs, a thermostat controller and washing machine. These two fundamentally different devices were chosen, as they give the reader a broad understanding of the process involved and the power of the SID tool.

The generation of SIDs follows the four stages outlined below:

1. Identification of core feedback
2. Determining the type of visualisation required
3. Capture/Generation of assets
4. Implementation of SID macro patches

7.4.1 Thermostat Controller

Within the context of smart systems and smart home research, the thermostat controller is a device receiving much attention. With the need to cut carbon emissions, research into the area of eco homes and energy reduction is increasing [14, 21]. Following the stages outlined above, the first stage of producing a SID is the identification of the core feedback provided by the physical device.

For this case study the thermostat controller will provide two simple feedback mechanisms: the current time and a temperature read out. This was based on the information available to the author from the thermostat controller used within their own home. With the feedback decided, the type of visualisation required needs to be designed along with the capture or generation of assets. How feedback is visualised and what type of assets should be captured or generated is part of the creative process. If an existing visualisation or device is being produced then the complexities and aesthetics of the actual device need to be determined. A designer must address aesthetical issues such as the age of the device, the type of display used and the control paradigm used (e.g. knobs, button, switches). If a brand new type of visualisation (or device) is being produced, assets will most likely have to be generated. The generation or capture of assets will depend on the type of visualisation. Should the assets, for example, be based on video? Will photographs be sufficient? Is the device available locally to be captured? Can an image from the internet be used? These types of questions will arise with the production of any new SID and the answers are device specific.

For this case study, to demonstrate the speed with which a simple device such as a thermostat controller can be produced, the type of visualisation will be based on an existing thermostat controller, similar to the one used within the authors home. This is a standard digital thermostat for controlling the heating and hot water.

Figure 7-12 shows a photograph of a thermostat controller being sourced from the internet and the subsequent modification of the photograph to remove the static time and temperature readout. Once these have been removed, dynamically controlled versions can be composited onto the photograph in real-time using Quartz Composer.

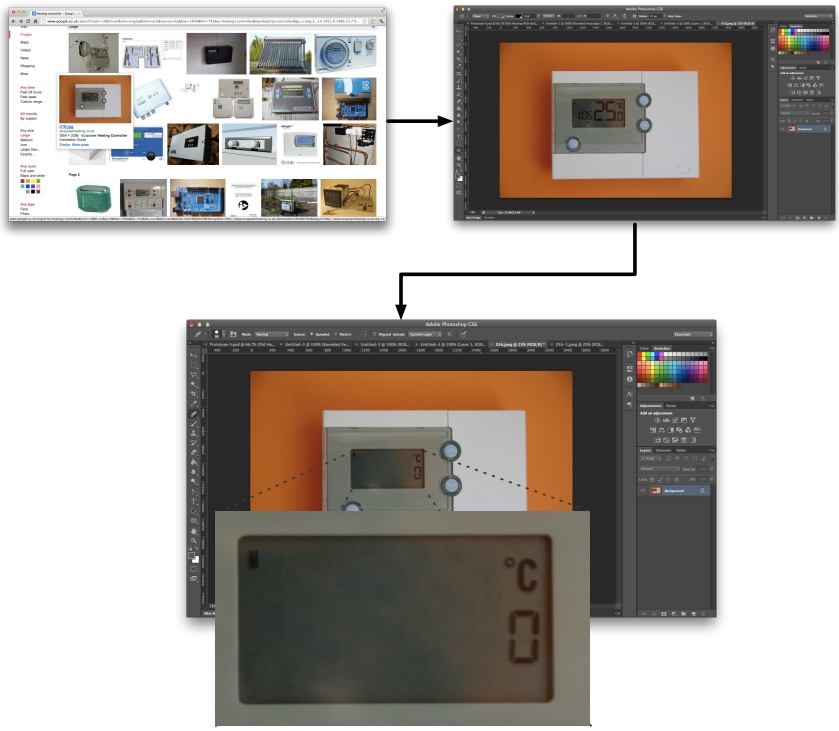

Figure 7-12 - Capture and Modification of Assets for Thermostat SID

All basic functionality for a thermostat controller has been provided through the SID macro patch library. From this library, four core macro patches will be used. These bespoke macro patches, created as part of this thesis, are *MIDI Controller, LCD Number, LCD Clock* and *Image Display.* Additional patches such as *Number* will also be used to provide inputs to the four core patches. Figure 7-13 shows the search within the library for *LCD* related patches, and the subsequent adding of the patch to the workspace.

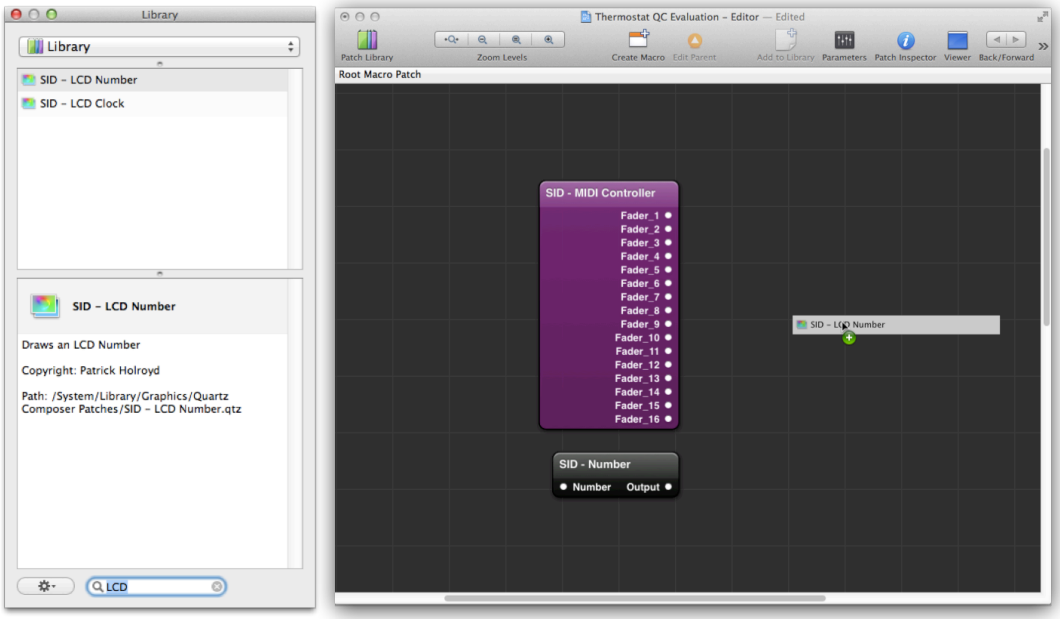

Figure 7-13 - Use of SID Macro Library in Quartz Composer for Development of Thermostat Controller SID

With the use of preformed SID macro patches, the complex logic is further hidden from the user. This provides greater accessibility for designers, many of who may not have a programming background. This was demonstrated by Carter et al. [102], who found that a significant percentage of designers do not like to programme. Basing the production of SID on non-traditional programming paradigms enables designers to focus on the creative aspects of SID production, such as what functionality of the device will be transferred from a physical world to a virtual one.

Figure 7-14 shows how SID macro patches are connected together within Quartz Composer. The temperature readout of the thermostat needs to be controlled by a wizard. Connecting the SID *LCD Number* patch to the SID *MIDI Controller* the number being displayed can be varied. When the fader is at 0% the number 0 will be displayed, at 100% the number 255 will be displayed.

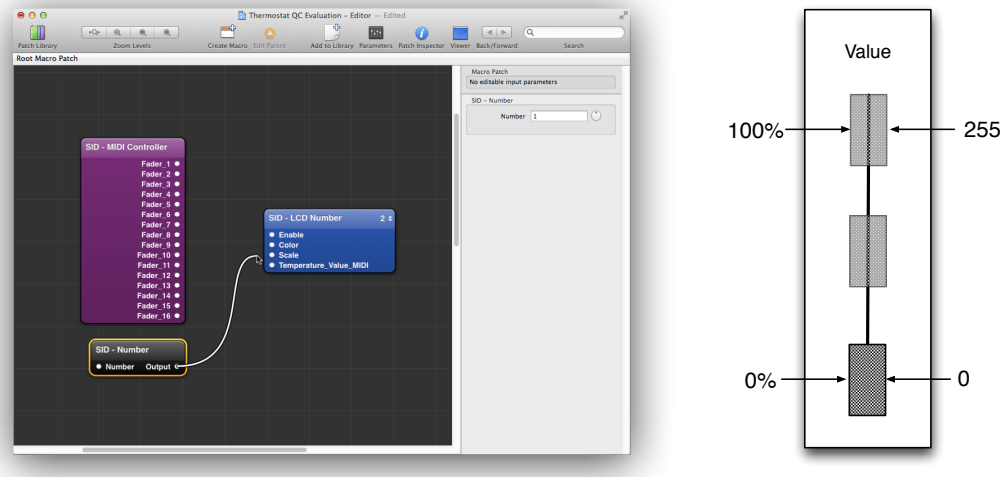

Figure 7-14 - Connecting SID Macro Patches Together and to a Digital Audio Mixer

Figure 7-15 shows the different components of the final temperature controller matched to their corresponding SID macro patches. The use of three SID output patches and a small number of input patches are required. The development of SID thermostat controller took approximately 20 minutes. As previously discussed, however, once complete, the thermostat can be instantly drawn upon for future evaluations.

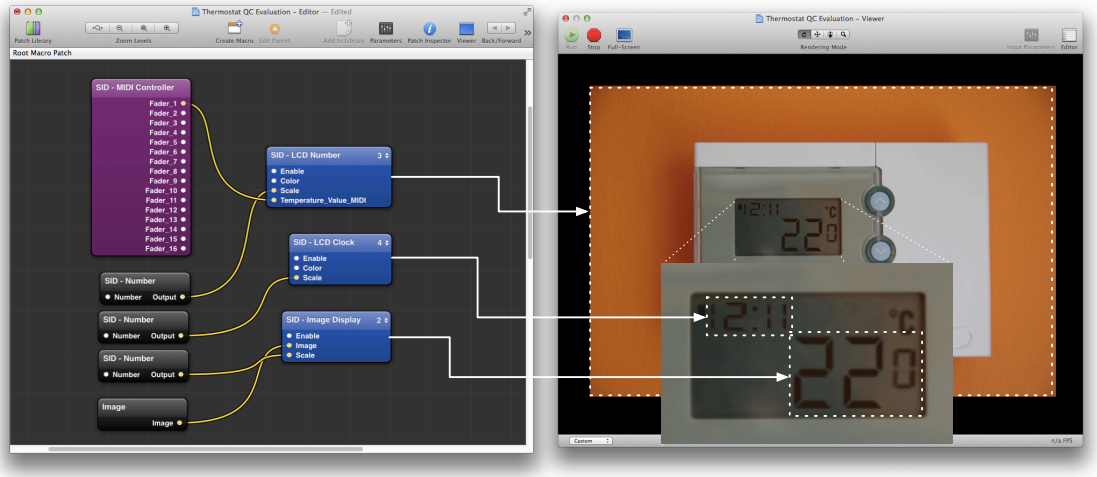

Figure 7-15 - SID Macro Patches and Corresponding Output

7.4.2 Washing Machine

The development of a washing machine is more complex, however, as with the temperature controller the type of feedback implemented will be based on an actual device. Analysing the device and considering its use within a domestic environment, the core feedback provided by a washing machine can be reduced to cycle type (including the temperature), spin speed and whether is it running or not. This is not to say that other feedback is not available, or that other devices offer different feedback, but rather when looking at a washing machine this is the core common feedback provided.

With the core feedback identified, the type of visualisation needs to be established. One of the basic and most obvious signs that a washing machine is on and working, or even the degree it is through it's cycle, is the physical motion of the device and the noise which is made during operation. For horizontal orientated washing machines (with a glass door) the visual appearance of the washing moving informs the user that the device is on and washing. If this can be replicated, it can be used to inform the subject when the machine has been started or stopped and what stage of the cycle it is in. The use of a digital visualisation and the replication of various LEDs, can further enhance the feedback provided.

By recording a real washing machine in motion, all the above feedback can be captured and composited with additional SID macro patches, thereby creating a complete controllable washing machine simulation.

Figure 7-16 - Video Capture and Editing of Real World Washing Machine

Figure 7-16 shows the recording of a washing machine in motion with the use of a smartphone. Setting the smartphone up to capture the washing machine *front on* gives the best angle for the final simulation. Approximately 5 minutes of material was captured with the entire capture process taking approximately 10 minutes. The raw footage was then trimmed to provide a piece of video 30 seconds long with the drum spinning at a medium speed always in the same direction. The trimming was completed using QuickTime Player X, a standard video application shipped with Apple Macs. Using QuickTime also enables the footage to be exported in H264 – an appropriate format for Quartz Composer. It was determined that the H264 format is optimal, based on the results of experiments with other video formats that resulted in a reduced performance of the system and jittery video playback.

Figure 7-17 shows the connection of the MIDI control interface to a SID *Video* patch. This macro patch enables the control of video using the fader control paradigm utilised as part of the SID tool. Play/Pause/Reverse can be controlled using a single fader – when located in the middle of the fader range the video will be paused, towards the top

of the range the video will play forward and towards the bottom of the range the video will play in reverse.

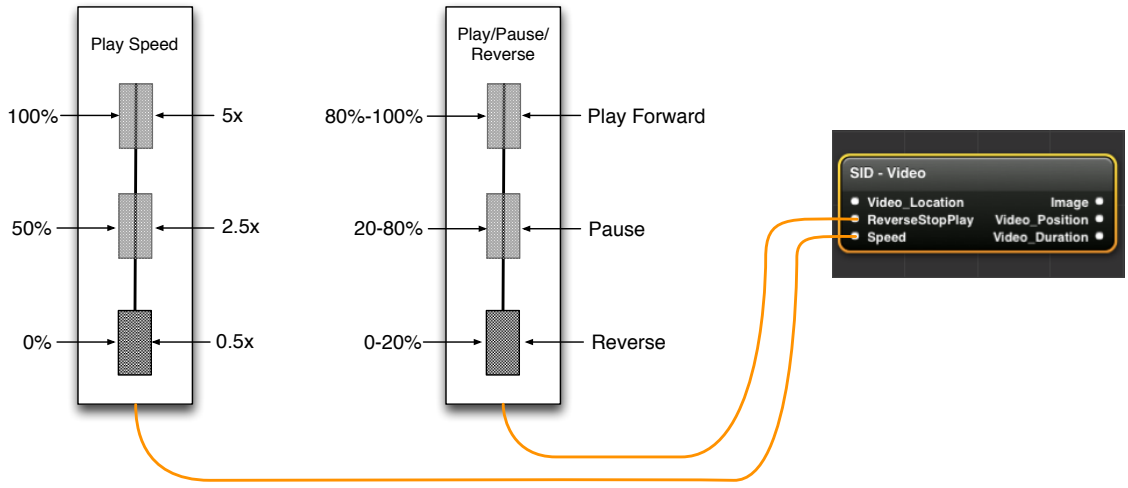

Figure 7-17 - Connection of the MIDI Control Interface to a SID Video Patch

A separate fader controls the speed of playback – the maximum playback speed is 5x the original, the minimum is 0.5x. Using these two faders the washing machine sequence can be paused – showing the device is not active, 0.5x – slow spin, 1x – medium spin and 5x – fast spin. Furthermore, because the video capture included audio, various clips can be played into the evaluation space in addition to the video simulation to provide a more convincing experience.

Figure 7-18 shows the SID macro patches required to implement the washing machine visualisation. In addition to the *Video* macro patch are the *Crop*, *Colour Controls, Image Display, MIDI Controller* and *LED* patches. Figure 7-19 show an example output of the *LED* patch used to visualise part of the control panel on the washing machine. Using a number of the *LED* patches enables the formation specific variable settings, e.g. cycle type.

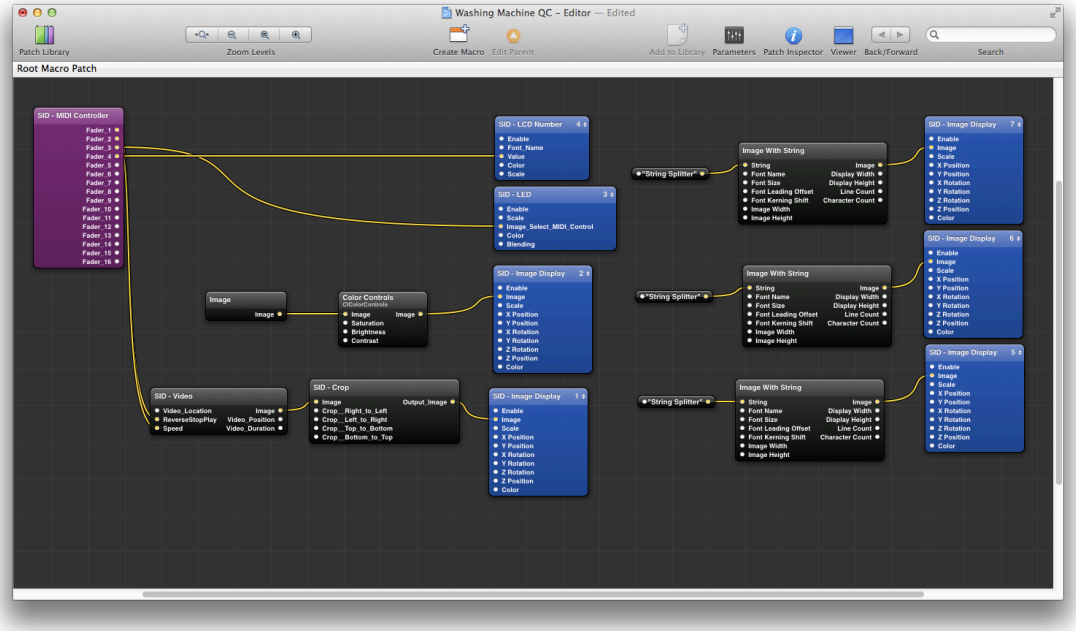

Figure 7-18 - SID Macro Patches Required for Washing Machine Visualisation

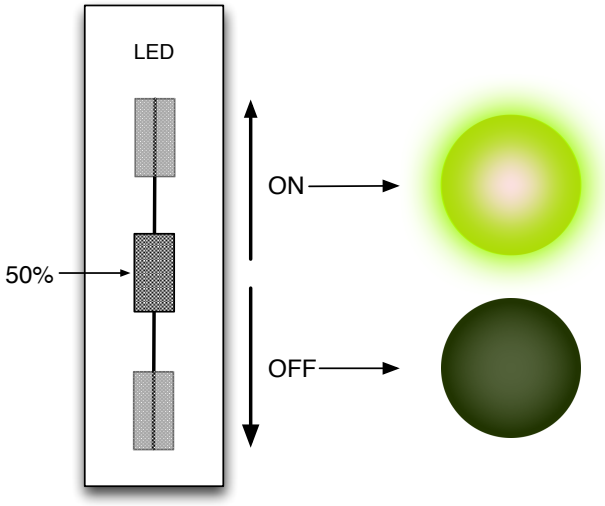

Figure 7-19 - SID LED

This case study of the SID tool shows the process undertaken by a user experience designer in production of devices commonly found within a domestic environment. The library of SID macro patches gives accessibility for designers when producing SIDs with Quartz Composer. Using the tool, devices commonly found within a domestic

environment can be simulated and manipulated in real-time by a wizard. Although based on video, accurate representations of devices can be formed, which can be used within the RME, thereby creating a pseudo smart environment for the evaluation of smart home control interfaces. Figure 7-20 shows a prototype being evaluated in the RME with the use of a SID.

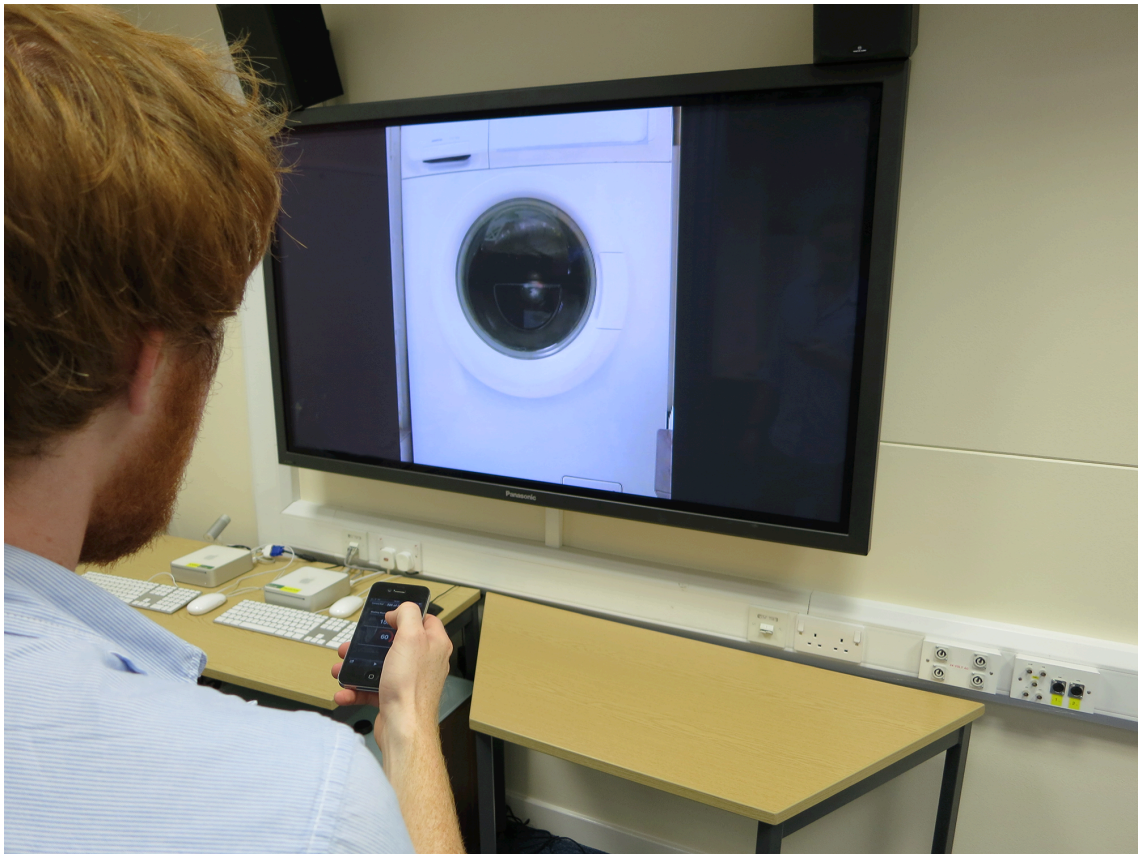

Figure 7-20 - Prototype Evaluation with SID in the RME

7.5 Standalone Use of the RME

At the time of writing the RME has been installed at the University of Sussex for over a year and in that time it has been utilised by a number of different researchers for various experiments. Using empirical feedback gathered during these experiments as well as the

author's personal data relating to reliability, the environment will be discussed and analysed.

In order to understand whether the RME is indeed usable, an empirical evaluation of the environment has been on-going throughout the past year. The results of this evaluation are based on observations by the author relating to the interactions between the users and the system, as-well-as questions presented to the users regarding their experiences. One group in particular will be focused upon, as it is this group that utilised the system to its full extent. The group in question were a number of researchers from the Open University (OU) that required the space to conduct user experience experiments.

The group has no prior knowledge of the RME and were therefore an ideal case study to assess the speed at which researchers adapted to the system and the extent of their success with the experiments. Their experiment involved connecting the output of a laptop to the RME and projecting a game onto a large touch screen panel. A number of subjects would then interact with the game via the panel, and the game output and subject interaction would be recorded by the RME.

Figure 7-21 shows the setup used for the experiment. This was determined and implemented by the researchers (initial setup took approximately 30 minutes) after they had undergone a training exercise. The training exercise lasted for approximately 15 minutes and involved an explanation of how to use the RME, in particular, how to connect devices to the system, set them up using the Visual Matrix software and setup the recording.

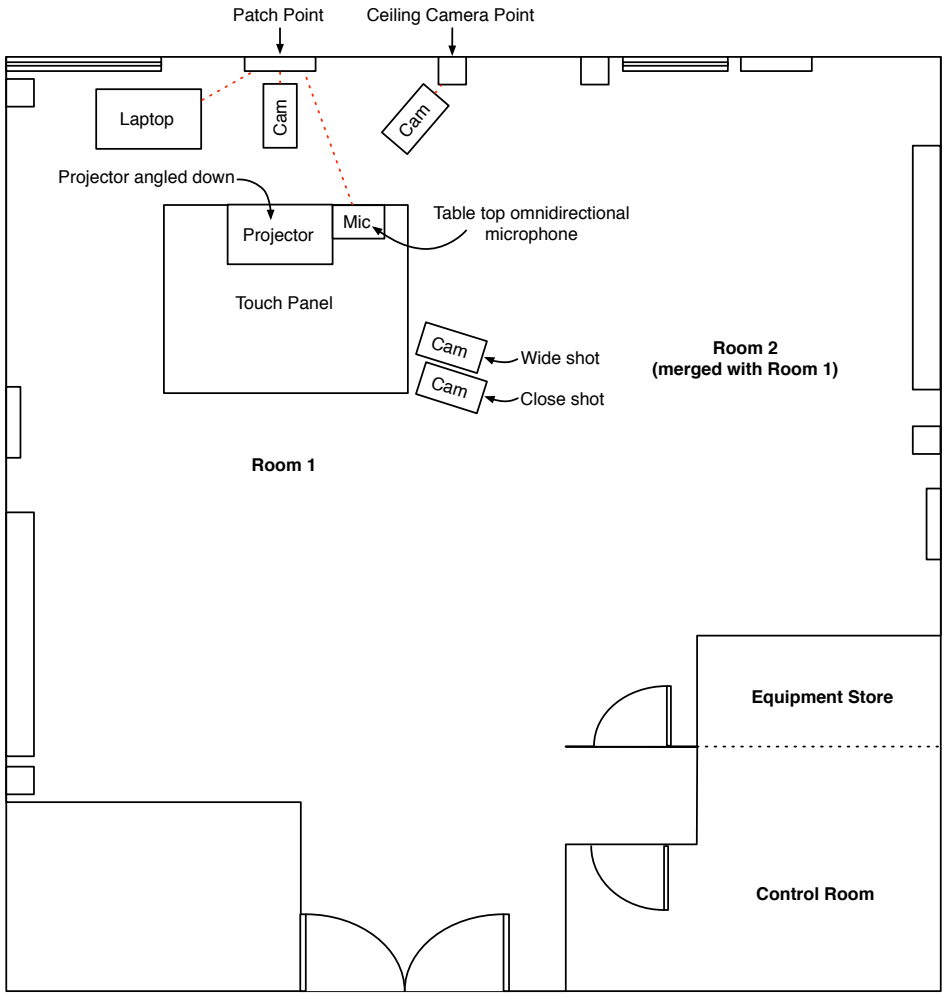

Figure 7-21 - Setup of RME for Case Study Experiment

Once setup, the researchers located themselves in the control room to observe the experiments. Numerous experiments were conducted over the period of a week and within that time additional support was required twice. The first was due to a failure with the recording software and the second was a simple query regarding the location of the captured video files.

When asked about the usability of the space the users responded positively. They expressed their pleasure with the speed at which the environment could be setup and configured. They felt they had received adequate knowledge of the system through the relatively short training session.

Having the researchers setup the space in approximately 30 minutes demonstrates the ability of the environment to be rapidly configurable, with the potential to support multiple simulation models, in this case a gaming environment (both requirements of the RME). The ability to quickly integrate their laptop computer to the system and record the game in real-time, as well as the subjects playing the game, also demonstrates the extensibility of the system.

In regards to the technical problem, this was due to a failure of the 3rd party capture software installed within the space. Having based the RME around the use of video and audio, specific components were used that would increase the reliability of the system. For example, the video matrix and audio mixer are devices that are built to broadcast quality standards and both are designed to perform over extremely long periods without failure.

Ultimately, however, problems will arise and with a year of use, technical issues have occurred. The video capture system is one area in particular that caused problems. As discussed in section 4.2 the capture machine is built on a Mac Pro with multiple video input cards and an off-the-shelf software solution (the capture machine integration within the RME can be seen in Figure 7-22). This software layer can at times be unstable causing the underlying OS to kernel panic. This is an issue with the 3rd party software that can be solved with additional development. The software is being supported by sufficient hardware and throughout all tests it ran well. An analysis of possible alternatives is currently on going; however, it is an issue that has been identified as needing a timely resolution.

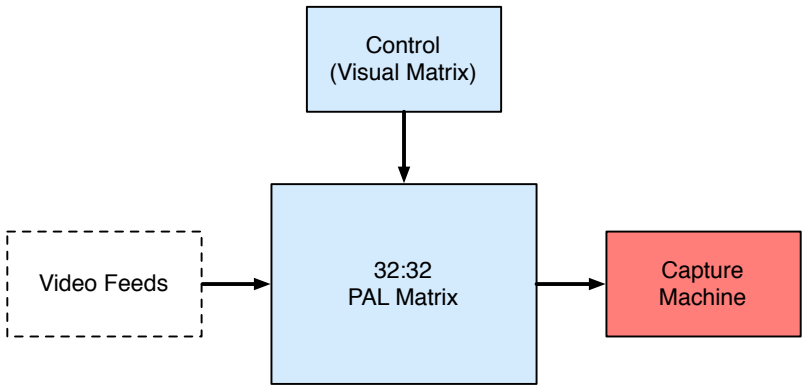

Figure 7-22 - Capture Machine Integration with RME

Since the installation of the RME, the Visual Matrix bespoke software, developed as part of this thesis, has been running continually. Throughout this time the software has performed as expected without significant failure. Users have reported that they feel confident with the systems and software and expect them to be reliable. Besides the issues with the capture machine, the RME has been reliable. Further to this particular set of experiments, users have followed the general theme of being satisfied with the RME. They have found the environment highly configurable and capable of capturing extensive amounts of data.

7.6 iProto Extension Layer

The production of the prototype outlined in section 7.2 has demonstrated the abilities of the iProto toolchain for round-tripping smartphone interface designs for smart home control. However, it has also demonstrated a disparity between the level of feedback provided by the SID tool and feedback from the iProto toolchain. Although iProto does allow feedback to be implemented, through the use of more windows, for wide ranging or complex feedback the process can, at times, be cumbersome to implement. For example, the thermostat controller outlined in section 7.2 required the temperature of the heating to be fed back to the user. It was easy to implement a single degree change up or down; however, a change of tens of degrees is unrealistic. For a full implementation, there would need to be over 250 different windows in use, each representing a single digit of temperature. To overcome this limitation, a means of fusing together the level of visual feedback provided by the SID tool, with the high fidelity prototyping and interaction provided by the iProto toolchain is proposed. In this way, feedback can be fed to the user through both SIDs and through prototype interfaces all from the same control interface and source system. Figure 7-23 shows a high-level overview of the proposed solution.

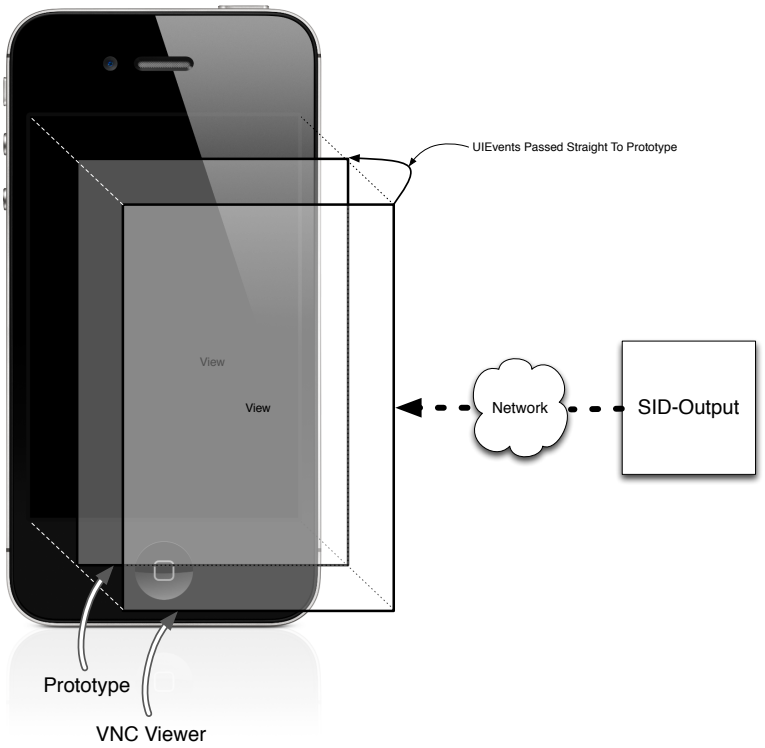

Figure 7-23 - Fusing iProto with SID Output

The technique is based on that developed by Jackson et al. [169]. It uses a VNC client to enable an image residing on a remote computer to be displayed on a portable handheld device. However, where as Jackson et al. used the technique to enable complex real-time modelling to appear on a handheld device, the iProto extension layer will use it to enable real-time SID outputs to be displayed on a smartphone device. This output will not replace the prototype being displayed; rather, it will be used to augment the prototype already deployed onto the device, providing a mobile version of the SID output that can be controlled by a wizard.

Figure 7-24 shows a screen from the prototype produced in section 7.2 augmented with the VNC SID layer. This can be controlled using the audio mixer based control paradigm, enabling a wizard to feedback information to the subject in real-time depending on their interactions.

Figure 7-24 - Prototype Augmented with VNC SID Layer

The layer will not provide interactive control with the subject, as all UI events will be passed directly to the prototype layer residing underneath. Using this technique, complex feedback can be delivered without the need to develop many windows within iProto. Figure 7-25 shows the VNC SID layer with the AirPlay feedback loop.

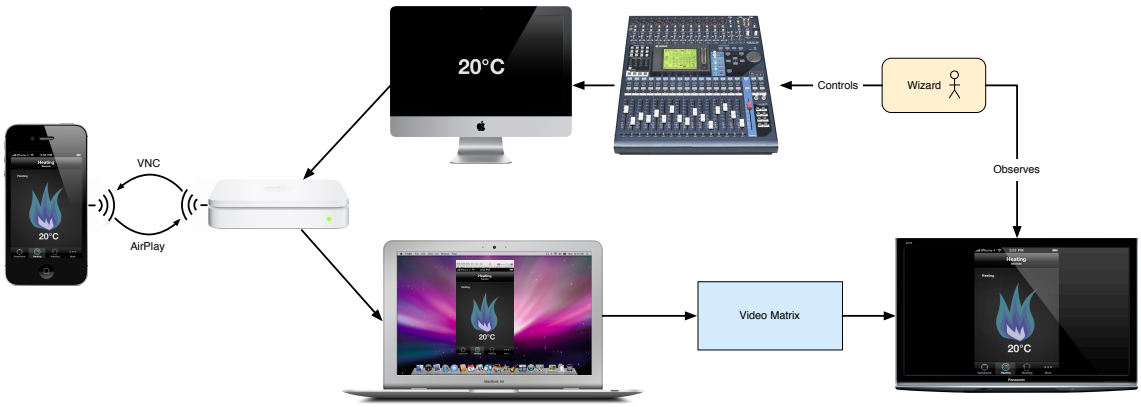

Figure 7-25 - Overview of VNC SID Layer with AirPlay Feedback Loop

7.7 Summary

This case study has shown the ability of the framework in facilitating the design and evaluation of smartphone interfaces (the preferred smart home control mechanism – as outlined in section 2.4.5). The study has discussed the use of the iProto toolchain for the round-tipping of smartphone interfaces and has demonstrated the efficient process involved with both major and minor interface modifications. It has also demonstrated the use of the SID tool in the production of simulated devices. In particular the accessibility provided for designers when producing SIDs with Quartz Composer and the ability of the tool to provide comparable feedback to the use of physical functioning devices. A discussion of the RME in a standalone context was exhibited through its use by a 3rd party. Specifically it has shown the configurability of the environment, the ease of setup and the efficient process involved with capturing user centred evaluation experiments.

Finally, a notable addition to the framework is introduced as the iProto Extension Layer. Following limitations identified during the case study of a disparity between the feedback provided by the SID tool compared with the iProto toolchain, the iProto Extension Layer proposes fusing together the iProto toolchain with SIDs. With this, a high-level of feedback can be provided through both SIDs and prototypes.

Chapter 8

Conclusions and Further Work

This chapter concludes the work carried out in this thesis and discusses the fulfilment of the original aims. These aims are first summarised, then individually assessed against the developed framework. The use of the various framework components developed for design, prototyping and evaluation of mobile interfaces for domestic environments is outlined. Finally, a section on further work will focus on how the framework can be extended with a consideration to the home of the future.

8.1 The Smart Home

This thesis begins by asking the question *why is my home not smart?* This led to the identification of four main issues preventing the adoption of smart homes that can be divided into four main areas: usability, retrofitting, cost and interoperability. Based on the available literature, work quickly centred on a main area preventing adoption: usability. In tackling the lack of good usability in smart homes this thesis proposed the design and development of a framework that could be used to facilitate prototyping and evaluation of smart home control interfaces. The framework proposes to achieve the following aims (discussed in section 3.5.1 – these aims have been paraphrased for increased clarity):

1. Provide a reconfigurable space for observation and capture of user centred research.
2. Support the use of the Wizard of Oz technique via dedicated simulation and communication tools for subject interaction.
3. Provide a means of producing prototypes for the current generation of smart home user interfaces - specifically the smartphone.
4. Improve the distribution of smartphone interface prototypes for evaluation.
5. Support researchers, specifically by satisfying the needs of the designer.
6. Facilitate the iterative prototyping and evaluation process for preferred smart home user interfaces.

The three major areas relating to this framework; smart home control, usability and simulation, are discussed in chapter 2. Included in this chapter are a number of techniques and processes that heavily influenced the designs, techniques and methodologies developed as part of the thesis. The basic usability principles are, for example, taken from those outlined for desktop computing. For although smartphones include various forms of interaction and control not traditionally available to desktop computers (e.g. gestures), the same underlying principles of usability can be adapted and built upon for the current generation of smart home control devices. The Wizard of Oz technique is another example of building upon previously developed techniques. This was discussed as part of the EasyLiving project, but it has also been utilised by many researchers dating back over the past twenty years [140].

Using a diverse set of techniques and processes influenced from the current literature and developed as part of this thesis, helped fulfil the aims and the development of the framework. Each individual aim is outlined below and is then individually assessed against the developed framework.

1. Provide a reconfigurable space for observation and capture of user centred research.

This aim was the premise for the design and development of the novel Reconfigurable Multimedia Environment (RME) [37, 38]. Using this environment researchers have been provided with a space which can be used to observe user centred research experiments and capture empirical quantitative and qualitative data. The environment has been designed as a generic space that can then be tailored to the needs to the user. The tailoring process includes bespoke and novel components that enable it to be configurable at speed – in many cases in real-time (e.g. Visual Matrix).

2. Support the use of the Wizard of Oz technique via dedicated simulation and communication tools for subject interaction.

The support of the Wizard of Oz technique has been provided through both the novel Simulated Interactive Devices (SID) tool [39] and the Extensible Feedback Mechanism (EFM). The SID is a video-based tool for the production of simulated devices normally found within the home. These simulations can be controlled by a wizard using a novel parallel video control paradigm. A bespoke software architecture and application framework enables two-way communications between the subject and the wizard with the use of different control GUIs.

3. Provide a means of producing high fidelity prototypes for the current generation of preferred smart home user interfaces, specifically the smartphone

This aim has been satisfied through the development of the novel iProto toolchain. This toolchain enables the rapid development of high fidelity prototypes for the current generation of smartphones. It provides support for gestures and for the deployment of prototypes onto an end device.

4. Improve the distribution of smartphone interface prototypes for evaluation

Included within the iProto toolchain is a cloud-based distribution system. This provides support for near instantaneous distribution of prototypes to end devices. Because it is cloud-based these end devices can be updated wirelessly anywhere in the world.

5. Support researchers, specifically by satisfying the needs of the designer.

The framework has been designed to conduct research. The configurability of the RME and the generic nature in which simulations and prototypes can be produced enable researchers to explore and evaluate a wide gamut of problems and solutions. This can be done without the need to have an understanding of programming or any underlying knowledge of the framework components. The integration of the iProto toolchain with the 3rd party graphics applications is a novel example of a smartphone prototyping tool fitting into an existing application pipeline, further supporting the needs of the designer.

6. Facilitate the iterative prototyping and evaluation process for preferred smart home user interfaces

This broad aim has been satisfied through the implementation of the novel framework. Through the use of the RME, SID and iProto, the framework facilitates the iterative prototyping and evaluation of smartphone interfaces; the preferred smart home user interface. This has been demonstrated through a case study (see chapter 7).

8.2 Utilising a Framework for Design, Prototyping and Evaluation of Mobile Interfaces for Domestic Environments

Using the framework researchers and designers are empowered with the necessary and previously unavailable tools to create the next generation of smart home control interfaces. Furthermore, the ability to evaluate such interfaces in the RME deployed at the University of Sussex gives the University a novel and cutting edge facility for conducting user centred research.

The framework is unique in its attempt to aggregate the disparate areas that are required to provide researchers with the instruments and tools needed to have a meaningful effect on the status of smart home usability. From this motivation, novel systems, applications and methodologies have been produced. It was never the intention of this thesis to prescribe new interfaces that have been deemed valid or correct, but rather to promote and facilitate the creation of intuitive interfaces through an enhanced prototyping techniques and more accurate and reliable evaluation data.

By providing support through the developed framework, a greater degree of usability, functionality and user satisfaction can be promoted. Without its support, smart home user interface research could become stagnant and the availability of intuitive, easy-to-use interfaces would be hindered.

8.2.1 Observing, Capturing and Simulating a Smart Home Environment with the RME and SID tool

This thesis presents the Reconfigurable Multimedia Environment as a novel space for conducting user centred research. The RME has primarily been designed to capture data relating to empirical, user-based, smart home research. It addresses the current need for a simulated smart home environment that is capable of being configured in real-time without any knowledge of the underlying hardware systems. This enables researchers to quickly and easily set-up an infinite range of experiments and further our understanding of smart home user interfaces. Although this thesis focuses on UI improvements facilitated by the RME, it can be used in various types of user-based research projects.

It is also designed to be extremely configurable when being initially setup and during scenario execution. The majority of the capture sensors can be moved and relocated to different areas within the space. Additional sensors can also be temporarily setup and connected directly to the system.

Basing the RME primarily on video provides a higher fidelity data set over more traditional capture methods (e.g. note taking). The control and manipulation of the various video feeds was paramount to the success of the environment. This required a new approach and the creation of a new visual routing paradigm. This materialised in

the development of Visual Matrix, a bespoke software application layer that enables real-time visual routing of video feeds.

The RME is also a platform for both the Simulated Interactive Devices (SID) and the EFM tools. The tools are built to integrate with both the underlying hardware layer of the RME and the Visual Matrix software layer.

The SID tool utilises video to create simulations of devices commonly found within a domestic smart environment. The device simulations facilitate interaction using the Wizard of Oz technique. Using this technique a wizard can respond to a subject interaction creating a pseudo realistic experience for subjects and enabling researchers to collect data on user interaction. Real-time video manipulation is permitted via the use of SID macro patches (OpenGL-based processing units developed using Quartz Composer) and the development of a real-time parallel control mechanism. The SID tool is a unique tool that incorporates a number of bespoke features. When combined with the RME, the resulting work is both novel and original [37, 38].

The framework was envisaged as complete solution for smart home control and usability research. However, the unique abilities of the RME have accelerated it to use in many other, different, areas of research. This has been demonstrated by its use with the Open University (OU) research team. The ability to configure the space to the requirements of the researchers and the ease with which they perform experiments provide evidence for its diverse use and success.

8.2.2 iProto

This thesis also presents the novel iProto toolchain for prototyping smart phone user interfaces. The toolchain encompasses a re-designed prototyping pipeline that includes a bespoke desktop authoring application for creating rapid high fidelity interactive prototypes. These prototypes can be deployed onto end devices, via a cloud-based distribution mechanism and a dedicated mobile app.

The iProto toolchain is based on findings from the available literature (outlined in section 2.5) that the best way to improve smart phone user interfaces is to facilitate the round-tripping prototyping process. The inadequacies of the current available tools led

to the development of both the re-designed prototyping pipeline and corresponding tools.

Shortly after completing the development of the iProto toolchain details were published by Jørgensen et al. [168] defining a similar process to that used by iProto. This process is included in *Touch Application Prototype* [168] and it uses the same concept of organising graphical assets into different *windows* and then linking those different *windows* together. However, the major similarity between the two systems is between the two redesigned pipelines; both incorporate the use of a widely used graphical asset generation package and both enable a prototype to be distributed to an end device.

Although *Touch Application Prototype* reduces the distinction of the iProto toolchain, its publication further cements the issues surrounding the prototyping of smartphone user interfaces. Having an independent group of researchers working on the same problem further demonstrates the validity and existence of the problem. Moreover, having the independent group produce a similar solution further validates the iProto toolchain.

Although both systems are similar there are still a number of key differences between them that make them both novel. *Touch Application Prototype*, for example, is built entirely around Adobe Fireworks and although this allows links and assets to be created in the same application, it is very limiting. Graphics have to be produced using Adobe Fireworks and not, in the case of iProto, any application which can conform to a simple export specification. With *Touch Application Prototype* the benefit of using the built-in functionality of the mobile internet browser allows subjects to swipe between pages and manipulate images using pinch and spread gestures (outlined in section 2.5.5). The same benefit can, however, also be a limitation. By basing the tool on web technologies, the *Touch Application Prototype* is limited by the web browser's ability to render elements and respond to touch inputs. If the browser is incapable of supporting a new type of gesture or it is incapable of rendering a new image format, it can never be incorporated into the prototype. The use of the web browser also limits the type of feedback, which can be seen by the wizard. The use of the interaction overlay is severely limited due to reliance of an application that has not been designed for user-based evaluations.

Axure [170] has also very recently implemented a similar technique to *Touch Application Prototype.* Using web technologies a user can produce a mock-up app and, using the default power of a web browser, they can navigate and interact with the prototype. The Axure solution is an elegant way of producing prototypes and it does have benefits compared with the iProto. The solution devised by Axure does, however, have flaws. In the same way *Touch Application Prototype* has been built on web technologies, these limitations are centred on the need for a browser in facilitating functionality. The browser seriously hinders the feedback and logging information that can be obtained during evaluations. Additionally, Axure requires the designer to manually integrate their graphics with the companies' prototype authoring solution.

Contrasting this, the iProto toolchain has been developed using the native development environment of Mac OS X and iPhone, allowing any type of code to be executed. This enables the iProto Mobile App to show overlays of subject interaction and to log interaction performed by the subject. Both the *Touch Application Prototype* and Axure solutions also fail to provide the capacity to evaluate prototypes. Without the abilities of the RME and SID tool these alternative prototyping solutions can be considered incomplete as they do not facilitate the evaluation and round-tipping of interfaces. Furthermore, the benefits from fusing together the iProto the SID tool provides an increased level of feedback unmatched by both the *Touch Application Prototype* and Axure solutions.

The novel characteristics of the RME, SID and iProto toolchain are ideal for tackling the issues surrounding the usability of smart homes and with it, the issues with smart home adoption. Through the use of the framework researchers now have the means to further our understanding of smart home user interfaces and realise a future where the benefits of smart technology and smart homes are available and accessible to all.

8.3 Further Work and the Home of the Future

The framework presented in this thesis has delivered a new approach to designing, prototyping and evaluating smart home user interfaces. There is, however, continued scope for research within both the confines of the framework and the wider area of smart home control.

The reliance of the RME on PAL resolution video was a design decision taken based on a number of factors (including the abilities of capture equipment and financial considerations). Although it is sufficient for providing a high-level of quality and detail is has quickly been superseded by HD video and more recently 4K resolution video. A move to capture video at an increased resolution would further enrich the data gathered by the RME whilst also allowing for the virtualisation of cameras.

As shown in Figure 8-1, this involves cropping the video image back down to PAL resolution then moving the copping boundaries around. This creates a similar effect as a camera moving to keep a particular object or person within the confines of the camera's viewing angle. Using this technique would reduce the number of cameras required and would further virtualise the remaining physical elements of the RME.

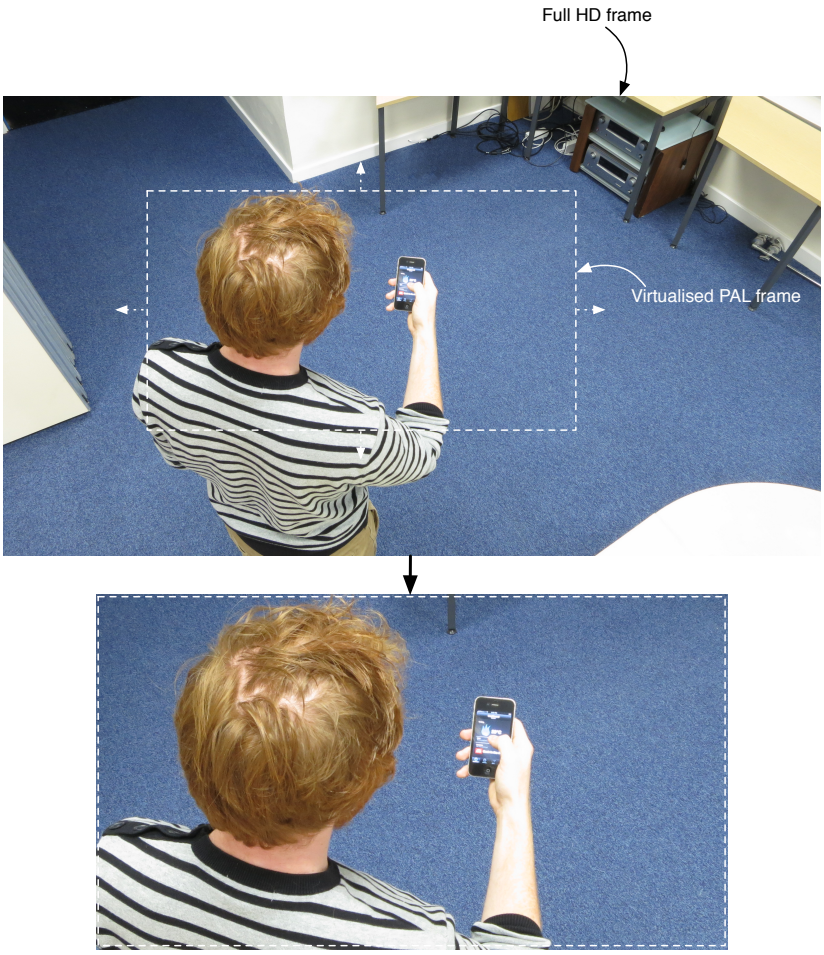

Figure 8-1 - Virtualised Camera

A movement towards HD video would also enable the automatic setup of cameras. The current analogue system makes its difficult to determine when a camera is connected to the RME. Human interaction is required with the Visual Matrix software layer and the manual generation of virtual camera and corresponding selection of connection points. Alternatively HD is digital, meaning it is relatively simple to determine when a device is connected to any of the connection points located around the space.

Further work relating to the iProto toolchain should centre on the ability of the iProto Mobile App to run native code on a smartphone device. Basing the toolchain on static graphical assets enabled tight integration with current app workflow methodologies, decreasing the effort and additional knowledge required to produce prototypes.

However, the advantages of native development allow for the creation of off-the-shelf UI components, which could be quickly incorporated into prototypes thereby providing a more realistic user experience. Further work on integrating the toolchain with other types devices (e.g. tablet computers) would also be an interesting useful addition to the established work.

The home of the future may, on the surface, look like the home of today. If the ultimate vision of ubiquitous computing is forthcoming it may even look like the home of yesterday. Why have a television in the corner of the room if your wallpaper is capable of displaying a movie? Ultimately the way in which we interact and control our home will change. As more devices are connected together creating the *Internet of Things* the challenge will be how to control all these devices. As this thesis has discussed, the control of the smart home will, in the near future, be dominated by the smartphone. However, the technical abilities of natural language processing and affective computing offer the next revolution in human computer interaction. Within this field there are two key control paradigms, which may provide a novel and improved experience for smart home control; full body gestures and voice control.

Full body gestures are an interpretation by a computer based on the actions and movements of the human body. Using a library of possible gestures, controlling a home can be accomplished by simply making the relevant gesture. This type of technology has already been integrated into computer games consoles where it has been very successful. Integrating the technology into the home, however, would present a number of challenges, specifically the additional need for context. A games console is a singular unit, when it is being controlled the player has to have their attention and focus on the game. Controlling a home is different. As discussed in section 2.4 home control will form into patterns or chains with the user potentially multitasking when performing actions. Gestures can easily interrupt those patterns and can make multitasking difficult. Is it easier to use a switch (whether physical or virtual) to set the oven temperature than it is to use a gesture? In relation to smart home control, gestures may find a place with simple generic tasks. For more complex issues voice control may be the answer.

The use of voice control could offer a complete revolution in the way we interact with our home. The potential for this type of interface is already being seen in the current generation of smartphones. *Siri* and *Google now* are both intelligent personal assistants released for the iPhone and Android phone respectfully. These applications use a natural language user interface to answer questions, make recommendations and perform actions.

If these types of control paradigms do become the focus of increased research, it will further allow the abilities of the RME and SID tool to be used and showcased. As demonstrated in the case study, their capacity is not limited to smartphone-based control research.

The home of the future *will* be controlled differently from the home of today. Developing the novel tools that allow a greater understanding to the usability of smart homes will enable a future where smart homes are available and accessible to all.

References

1. Aldrich, F.K., J. Barlow, K. Cheverst, K. Clarke, G. Dewsbury, T. Diggins, D. Frohlich, L. Hamill, R. Harper, T. Hemmings, J. Hughes, M. Jokinen, A. Karsenty, R. Kraut, S. Leppänen, A. MacLean, S. Peters, J. Pycock, D. Randall, M. Rouncefield, B. Shatwell, J.D. Strain, A. Taylor, P. Tolmie, and T. Venables, Inside the smart home, R. Harper. 2003: Springer.
2. Cook, D. and S.K. Das, Smart environments. Wiley series on parallel and distributed computing. 2005, Hoboken, New Jersey.: John Wiley & Sons, Inc.
3. Ricquebourg, V., D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge. The Smart Home Concept : our immediate future. in E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on. Year.
4. Helal, S., W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, The Gator Tech Smart House: a programmable pervasive space. Computer, 2005. 38(3): p. 50-60.
5. Kidd, C.D., R. Orr, G.D. Abowd, C.G. Atkeson, I.A. Essa, B. MacIntyre, E.D. Mynatt, T. Starner, and W. Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. in Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture. 1999: Springer-Verlag.
6. Yamazaki, T., The Ubiquitous Home. International Journal of Smart Home, 2007. 1(1).
7. Mozer, M.C., 12 Lessons from an Adaptive House, in Smart environments: technologies, protocols, and applications, D.J. Cook and S.K. Das, Editors. 2005, J. Wiley & Sons, Hoboken: NJ. p. 273- 294.
8. Intille, S.S., Designing a home of the future. Pervasive Computing, IEEE, 2002. 1(2): p. 76-82.
9. Harper, R., R. Banks, L. Barkhuus, J. Barlow, S. Bayer, B. Brown, D. Dunkle, J. Forlizzi, L. Hamill, J. Kaye, D. Kirk, S.C.R. Lewis, S. Lindley, W. Odom, T.C. Oliveira, and D. Randall, The Connected Home: The Future of Domestic Life. 1 ed, R. Harper. 2011, London: Springer-Verlag.
10. Holroyd, P., P. Watten, and P. Newbury. Why is my home not smart? in Proceedings of the Aging friendly technology for health and independence, and

8th international conference on Smart homes and health telematics. 2010, Seoul, Korea: Springer-Verlag.
11. Barlow, J. and D. Gann. A changing sense of place: are integrated IT systems reshaping the home? in Technological Futures, Urban Futures. 1998, Durham: University of Sussex, Science Policy Research Unit.
12. CORDIS. ICT Challenge 5: ICT for Health, Ageing Well, Inclusion and Governance. 2012 [cited 2012 07/08]; Available from: http://cordis.europa.eu/fp7/ict/programme/challenge5_en.html.
13. Aging Friendly Technology for Health and Independence, 8th International Conference on Smart Homes and Health Telematics. in ICOST. Year, Seoul.
14. Fischer, C., Feedback on household electricity consumption: a tool for saving energy? Energy Efficiency, 2008. 1(1): p. 79-104.
15. Wood, G. and M. Newborough, Energy-use information transfer for intelligent homes: Enabling energy conservation with central and local displays. Energy and Buildings, 2007. 39(4): p. 495-503.
16. Brush, A.J.B., B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, and C. Dixon. Home automation in the wild: challenges and opportunities. in Proceedings of the 2011 annual conference on Human factors in computing systems. 2011, Vancouver, BC, Canada: ACM.
17. Raad, M. and L. Yang, A ubiquitous smart home for elderly. Information Systems Frontiers, 2009. 11(5): p. 529-536.
18. Change, D.o.E.C. Smart meters: a guide. 2013 [cited 2013 26/04]; Available from: https://http://www.gov.uk/smart-meters-how-they-work - timeframes-for-installation.
19. Darby, S., The Effectiveness Of Feedback On Energy Consumption, F.a.R.A. Department for Environment, Editor 2006, Environmental Change Institute, University of Oxford.
20. Lu, J., T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. in Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems. 2010, Zurich, Switzerland: ACM.
21. Change, D.o.E.C. Smart Meters. 2012 [cited 2012 17/09]; Available from: http://www.decc.gov.uk/en/content/cms/tackling/smart_meters/smart_meters.aspx.
22. Min-Soo, K., K. Kyung Mi, and K. Hee-Cheol. A questionnaire study for the design of smart home for the elderly. in e-Health Networking, Applications and Services, 2006. HEALTHCOM 2006. 8th International Conference on. Year.
23. Pragnell, M., L. Spence, and R. Moore, The market potential for Smart Homes, in Joseph Rowntree Foundation2000, York Publishing Services Ltd: England.
24. Mitzner, T.L., J.B. Boron, C.B. Fausset, A.E. Adams, N. Charness, S.J. Czaja, K. Dijkstra, A.D. Fisk, W.A. Rogers, and J. Sharit, Older adults talk technology: Technology usage and attitudes. Computers in Human Behavior, 2010. 26(6): p. 1710-1721.
25. Graham, R., Group differences in attitudes towards technology among Americans. New Media & Society, 2010. 12(6): p. 985-1003.

26. Gurganious, D., M. Hatler, and M. Ritter, WSN for Smart Home: A Market Dynamics Report, 2008, ON World: San Diego, California.
27. Deffree, S., Revenue gain expected in 2010 consumer electronics market. Electronic News, 2010. 56(6): p. 3-3.
28. Davidoff, S., M. Lee, C. Yiu, J. Zimmerman, and A. Dey, Principles of Smart Home Control, P. Dourish and A. Friday, Editors. 2006, Springer Berlin / Heidelberg. p. 19-34.
29. Edwards, W.K. and R.E. Grinter. At Home with Ubiquitous Computing: Seven Challenges. in Proceedings of the 3rd international conference on Ubiquitous Computing. 2001, Atlanta, Georgia, USA: Springer-Verlag.
30. Alliance, W.-F. Wi-Fi Alliance. 2013 [cited 2013 26/04]; Available from: http://www.wi-fi.org/.
31. Alliance, Z. Zigbee Alliance. 2010 [cited 2010 03/03]; Available from: http://www.zigbee.org/.
32. X10.com. 30+ years in business, online since 1996! Pico Electronics 2012 [cited 2012 26/06]; Available from: http://www.x10.com/support/faq_privacy.html.
33. Echelon. Lonworks 2.0. 2010 [cited 03/03/2010; Available from: http://www.echelon.com/products/lonworks/.
34. KNX. KNX Standard. 2009 [cited 2010 08/03]; Available from: http://www.knx.org/knx-standard/introduction/.
35. Hamdhaidari, M., High LevelModelling And Simulation of Pervasive Multimedia Systems And Environments, in Informatics2009, University of Sussex: Brighton.
36. Jeong, K.-A., R.W. Proctor, and G. Salvendy, A Survey of Smart Home Interface Preferences for U.S. and Korean Users. Human Factors and Ergonomics Society Annual Meeting Proceedings, 2009. 53: p. 541-545.
37. Holroyd, P., P. Watten, and P. Newbury. Reconfigurable Multimedia Environment. in Proceedings of the 5th international conference on Ubiquitous and Collaborative Computing. 2010, Abertay, Dundee: British Computer Society.
38. Holroyd, P., P. Watten, and P. Newbury, Reconfigurable Multimedia Environment. Journal of Emerging Technologies in Web Intelligence, 2011. 3(5): p. 282-285.
39. Rivera, F., P. Watten, P. Holroyd, F. Beacher, K. Mania, and H. Critchley. Real-time compositing framework for interactive stereo fMRI displays. in ACM SIGGRAPH 2010 Posters. 2010, Los Angeles, California: ACM.
40. Weiser, M., The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev., 1999. 3(3): p. 3-11.
41. Weiser, M. and J.S. Brown, The coming age of calm technolgy, in Beyond calculation: the next fifty years1997, Copernicus. p. 75-85.
42. rwilliamson. Inside the Apple Lightning Cable. 2012 [cited 2013 06/05]; Available from: http://www.chipworks.com/blog/recentteardowns/2012/10/15/inside-the-apple-lightning-cable/.

43. Weiser, M. and R. Gold, The origins of ubiquitous computing research at PARC in the late 1980s. IBM Systems Journal, 1999. 38(4): p. 693.
44. Evans, D., The Internet of Things - How the Next Evolution of the Internet is Changing Everything, in White Paper2011, Cisco.
45. TOP500 List - November 2010 TOP500, 2010.
46. Gershenfeld, N., R. Krikorian, and D. Cohen, The Internet of things. Scientific American, 2004. 291(4): p. 76-81.
47. Economist, Punching the card, in The Economist2003, The Economist Newspaper Limited: London and San Francisco.
48. Atzori, L., A. Iera, and G. Morabito, The Internet of Things: A survey. Comput. Netw., 2010. 54(15): p. 2787-2805.
49. Ma, J., L.T. Yang, B.O. Apduhan, R. Huang, L. Barolli, and M. Takizawa, Towards a smart world and ubiquitous intelligence: A walkthrough from smart things to smart hyperspaces and UbicKids. International Journal of Pervasive Computing and Communications, 2005. 1(1): p. 53-68.
50. Hindus, D. The Importance of Homes in Technology Research. in Cooperative Buildings. Year.
51. Poulson, D., C.A. Nicolle, and M. Galley, Review of the current status of research on smart homes and other domestic assistive technologies in support of the TAHI trials, 2002, Loughborough University: Loughborough.
52. Park, S.H., S.H. Won, J.B. Lee, and S.W. Kim, Smart home – digitally engineered domestic life. Personal and Ubiquitous Computing, 2003. 7(3): p. 189-196.
53. Research Areas, Groups and Collaborators for the Aware Home Research Initiative. 2008 [cited 2008 30/03/2009]; Available from: http://awarehome.imtc.gatech.edu/research.
54. Meeks, G. Georgia Tech's Aware Home Research Initiative. [cited 2012 11/04]; Available from: http://www.cc.gatech.edu/fce/house/house.html.
55. Summet, J., G.D. Abowd, G.M. Corso, and J.M. Rehg. Virtual rear projection: do shadows matter? in CHI '05 extended abstracts on Human factors in computing systems. 2005, Portland, OR, USA: ACM.
56. Sanders, J.M. Sensing the Subtleties of Everyday Life. 2000 [cited 2012 20/08]; Available from: http://gtresearchnews.gatech.edu/reshor/rh-win00/main.html.
57. Orr, R. and G. Abowd. The Smart Floor: A Mechanism for Natural User Identification and Tracking. in Proc. of the 2000 Conference on Human Factors in Computing Systems (CHI '00). Year.
58. Gregory D. Abowd, A.F.B., Irfan A. Essa, Elizabeth D. Mynatt, Wendy A. Rogers. The Aware Home: A living laboratory for technologies for successful aging. in AAAI-02 Workshop “Automation as Caregiver. 2002, Canada: AAAI.
59. Saha, D. and A. Mukherjee, Pervasive Computing: A Paradigm for the 21st Century. Computer, 2003. 36(3): p. 25-31.
60. Helal, S. The Gator Tech Smart House. 2005 [cited 2012 11/04]; Available from: http://www.icta.ufl.edu/gt.htm - 1.
61. Yamazaki, T. Ubiquitous home: real-life testbed for home context-aware service. in Testbeds and Research Infrastructures for the Development of

Networks and Communities, 2005. Tridentcom 2005. First International Conference on. Year.
62. Sakagami, Y., R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. in Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Year.
63. Gartner Says Consumers On Track to Spend $2.1 Trillion Globally on Digital Information and Entertainment Products and Services in 2011, in Press Releases2011, Gartner: Stamford, Connecticut.
64. Economist, Apple's share price, in The Economist2012, The Economist Newspaper Limited: London and San Francisco.
65. Laavola, T., K. Haataja, J. Mielikainen, and P. Toivanen. Designing and implementing an intelligent X-10-enabled home: Studies in home intelligence. in Internet, 2008. ICI 2008. 4th IEEE/IFIP International Conference on. Year.
66. Dewsbury, G., B. Taylor, and M. Edge. The Process of Designing Appropriate Smart Homes: Including the User in the Design. in The 1st Equator IRC Workshop on Ubiquitous Computing in Domestic Environments, The School of Computer Science and Information Technology. 2001, University of Nottingham.
67. Hamill, L., Controlling smart devices in the home. The Information Society, 2006. 22(4): p. 241-249.
68. Brouwer-Janse, M.D., R.W. Bennett, T. Endo, F.L.v. Nes, H.J. Strubbe, and D.R. Gentner. Interfaces for consumer products: "how to camouflage the computer?". in Proceedings of the SIGCHI conference on Human factors in computing systems. 1992, Monterey, California, United States: ACM.
69. Nichols, J. and B.A. Myers. Studying the Use of Handhelds to Control Smart Appliances. in Proceedings of the 23rd International Conference on Distributed Computing Systems. 2003: IEEE Computer Society.
70. Barkhuus, L. and A. Dey, Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined, A. Dey, A. Schmidt, and J. McCarthy, Editors. 2003, Springer Berlin / Heidelberg. p. 149-156.
71. Crabtree, A., T. Hemmings, and T. Rodden. Pattern-based support for interactive design in domestic settings. in Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques. 2002, London, England: ACM.
72. Koskela, T., K. Väänänen-Vainio-Mattila, and L. Lehti, Home Is Where Your Phone Is: Usability Evaluation of Mobile Phone UI for a Smart Home, in Mobile Human-Computer Interaction – MobileHCI 20042004. p. 74-85.
73. Yuansheng, L. Design of the Smart Home based on embedded system. in Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD '06. 7th International Conference on. Year.
74. Rashidi, P. and D.J. Cook, Keeping the Resident in the Loop: Adapting the Smart Home to the User. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2009. 39(5): p. 949-959.
75. Nintendo. Wii. 2012 [cited 2012 11/04]; Available from: http://www.nintendo.co.uk/NOE/en_GB/wii_54.html.

76. Microsoft. Xbox 360 + Kinect. 2012 [cited 2012 11/04]; Available from: http://www.xbox.com/en-US/kinect.
77. Apple Inc. iPhone. 2012 [cited 2012 12/04]; Available from: http://www.apple.com/iphone/features/.
78. Nielsen, J. Remote Control Anarchy. Alertbox, 2004.
79. Koskela, T. and K. Vaananen-Vainio-Mattila, Evolution towards smart home environments: empirical evaluation of three user interfaces. Personal Ubiquitous Comput., 2004. 8(3-4): p. 234-240.
80. Lorenz, A., C.F.D. Castro, and E. Rukzio. Using handheld devices for mobile interaction with displays in home environments. in Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. 2009, Bonn, Germany: ACM.
81. Myers, B.A., J. Nichols, J.O. Wobbrock, and R.C. Miller, Taking Handheld Devices to the Next Level. Computer, 2004. 37(12): p. 36-43.
82. Suo, Y., C. Wu, Y. Qin, C. Yu, Y. Zhong, and Y. Shi. HouseGenie: Universal Monitor and Controller of Networked Devices on Touchscreen Phone in Smart Home. in Proceedings of the 2010 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing. 2010: IEEE Computer Society.
83. Cisco, Global Mobile Data Traffic Forecast Update, 2011–2016, in Cisco Visual Networking Index2012, Cisco.
84. HMB-TEC. HouseControl. 2012 [cited 20/08 2012]; Available from: http://itunes.apple.com/us/app/housecontrol/id346433698?mt=8.
85. Economist, Beyond the PC, in The Economist2011, The Economist Newspaper Limited: London and San Francisco.
86. Nichols, J. and B.A. Myers, Controlling Home and Office Appliances with Smart Phones. Pervasive Computing, IEEE, 2006. 5(3): p. 60-67.
87. Zivkov, D., B. Majstorovic, T. Andjelic, M. Davidovic, and D. Simic. Smart-Phone application as TV remote controller. in Consumer Electronics (ICCE), 2012 IEEE International Conference on. Year.
88. Nielsen, J., The usability engineering life cycle. Computer, 1992. 25(3): p. 12-22.
89. Philips. Philips MyRemote. 2012 [cited 2012 28/08]; Available from: http://itunes.apple.com/gb/app/philips-myremote/id426883783?mt=8.
90. BSkyB. Sky+. 2012 [cited 2012 28/08]; Available from: http://itunes.apple.com/gb/app/sky+/id301250225?mt=8.
91. Virgin Media. Virgin Media TV Guide. 2011 [cited 2012 28/08]; Available from: http://itunes.apple.com/gb/app/virgin-media-tv-guide/id449156074?mt=8.
92. CO.LTD, S.E. Samsung Remote. 2011 [cited 2011 14/07]; Available from: http://itunes.apple.com/gb/app/samsung-remote/id359580639?mt=8.
93. Pan, Y. and S.R. Young, Insights for the TV Interface from the Mobile Phone Interface. Journal of Usability Studies, 2009. 4(4): p. 166-177.
94. Insteon. Insteon. 2012 [cited 2012 28/08]; Available from: http://www.insteon.net/.
95. Berro, M. HomeControl. 2010 [cited 2012 28/08]; Available from: http://itunes.apple.com/gb/app/homecontrol/id366687584?mt=8.

96. Unger, R. and C. Chandler, A Project Guide to UX Design: For user experience designers in the field or in the making. 1 ed. 2009, Berkeley: New Riders.
97. Roduner, C., M. Langheinrich, C. Floerkemeier, and B. Schwarzentrub, Operating Appliances with Mobile Phones – Strengths and Limits of a Universal Interaction Device, in Pervasive Computing2007. p. 198-215.
98. Sauter, V.L. What is Prototyping? Prototyping 2012 [cited 2012 03/04]; Available from: http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html.
99. Nielsen, J., Iterative user-interface design. Computer, 1993. 26(11): p. 32-41.
100. Nielsen, J. Parallel & Iterative Design + Competitive Testing = High Usability. 2011 [cited 2012 06/06]; Available from: http://www.nngroup.com/articles/parallel-and-iterative-design/.
101. Norman, D.A. and S.W. Draper, User Centered System Design; New Perspectives on Human-Computer Interaction. 1986: L. Erlbaum Associates Inc. 526.
102. Carter, A.S. and C.D. Hundhausen. How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. in Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on. Year.
103. Lindholm, C., Mobile usability;how Nokia changed the face of the mobile phone, T. Keinonen and H. Kiljander. 2003: McGraw-Hill.
104. Arthur, C., Nokia's chief executive to staff: 'we are standing on a burning platform', in The Guardian2011, Guardian News and Media Limited: London.
105. Jaapleving. Touchscreen Hand Gestures V2. 2009 [cited 2012 15/09]; Available from: http://graffletopia.com/stencils/439.
106. Grossman, L., Invention of the Year: The iPhone, in Time2007, Time.
107. Solutions, A.S. Axure. 2012 [cited 2012 09/07]; Available from: http://www.axure.com/.
108. Likecool.com. Notepod. Gear, Office 2009 [cited 2012 22/03]; Available from: http://www.likecool.com/Notepod--Office--Gear.html.
109. UI Stencils. UI Stencils. 2011 [cited 2012 22/03]; Available from: http://www.uistencils.com/.
110. Blog*spark. Fireworks toolkit for creating iPhone UI mockups. 2009 [cited 2012 22/03]; Available from: http://blog.metaspark.com/2009/02/fireworks-toolkit-for-creating-iphone-ui-mockups/.
111. 320480.com. iPhone Interface PSD. 2008 [cited 2012 22/03]; Available from: http://320480.com.
112. graffletopia.com. iPad and iPhone Design. 2010 [cited 2012 22/03]; Available from: http://graffletopia.com/categories/user-interface.
113. Studios, B. balsamiq. 2012 [cited 2012 09/07]; Available from: http://www.balsamiq.com/.
114. teehan+lax. iOS 5 GUI PSD. 2012 [cited 2012 21/08]; Available from: http://www.teehanlax.com/downloads/ios-5-gui-psd-iphone-4s/.
115. Bolchini, D., D. Pulido, and A. Faiola, FEATURE: "Paper in screen" prototyping: an agile technique to anticipate the mobile experience. interactions, 2009. 16(4): p. 29-33.
116. Hennipman, E.-J., E.-J.R.G. Oppelaar, G.C.v.d. Veer, and B. Bongers. Rapid and rich prototyping: proof of concepts for experience. in Proceedings of the

15th European conference on Cognitive ergonomics: the ergonomics of cool interaction. 2008, Funchal, Portugal: ACM.
117. Houde, S. and C. Hill, What Do Prototypes Prototype? Handbook of Human-Computer Interaction, 1997.
118. Meskens, J., J. Vermeulen, K. Luyten, and K. Coninx. Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me. in Proceedings of the working conference on Advanced visual interfaces. 2008, Napoli, Italy: ACM.
119. Meskens, J., K. Luyten, and K. Coninx. Jelly: a multi-device design environment for managing consistency across devices. in Proceedings of the International Conference on Advanced Visual Interfaces. 2010, Roma, Italy: ACM.
120. groosoft. Blueprint Viewer. 2011 [cited 2012 22/03]; Available from: http://itunes.apple.com/gb/app/blueprint-viewer/id411622447?mt=8.
121. keynotopia.com. Keynotopia Templates. 2011 [cited 2012 22/03]; Available from: http://keynotopia.com/.
122. Adobe. Adobe Proto. Adobe Touch Apps family 2012 [cited 2012 04/04]; Available from: http://www.adobe.com/products/proto.html.
123. Soley, J. Refocusing our Touch App Development. Creative Cloud Team Blog 2012 [cited 2013 30/04]; Available from: http://blogs.adobe.com/creativecloud/refocusing-our-touch-app-development/.
124. Nielsen, J. Usability inspection methods. in Conference companion on Human factors in computing systems. 1994, Boston, Massachusetts, United States: ACM.
125. Coen, M.H. Design principles for intelligent environments. in Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence. 1998, Madison, Wisconsin, United States: American Association for Artificial Intelligence.
126. Lucente, M., G.-J. Zwart, and A.D. George. Visualization Space: A Testbed for Deviceless Multimodal User Interface. in Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments. Year: AAAI.
127. Badre, A.N., M. Guzdial, S.E. Hudson, and P.J. Santos, A user interface evaluation environment using synchronized video, visualizations and event trace data. Software Quality Journal, 1995. 4(2): p. 101-113.
128. Dix, A., J. Finlay, G. Abowd, and R. Beale, Human-Computer Interaction (3rd Edition). 2003: Prentice Hall. 789.
129. Kheir, N.A. and M. Dekker, Systems modeling and computer simulation. 1988, New York, NY, USA: Marcel Dekker, Inc.
130. Lehman, R.S., Computer Simulation and Modeling: An Introduction. 1977, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, Publishers.
131. Zeigler, B.P., Theory of Modelling and Simulation. 1976: Wiley-Interscience.
132. Shannon, R.E., Systems Simulation. The Art and Science. 1975, London, UK: Englewood Cliffs, EUA : Prentice-Hall.
133. John S. Carson, I. Introduction to modeling and simulation. in Proceedings of the 37th conference on Winter simulation. 2005, Orlando, Florida: Winter Simulation Conference.

134. Jacobson, I., M. Christerson, P. Jonsson, and G. Overgaard, Object-oriented software engineering;a use case driven approach, I. Jacobson. 1992: Addison-Wesley Publishing Company.
135. McConnell, S., Software Project Survival Guide (Pro -- Best Practices). 1997: {Microsoft Press}.
136. Lertlakkhanakul, J., J.W. Choi, and M.Y. Kim, Building data model and simulation platform for spatial interaction management in smart home. Automation in Construction, 2008. 17(8): p. 948-957.
137. Norbisrath, U., I. Armac, D. Retkowitz, and P. Salumaa. Modeling eHome systems. in Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006). 2006, Melbourne, Australia: ACM.
138. Armac, I. and D. Retkowitz. Simulation of Smart Environments. in Pervasive Services, IEEE International Conference on. Year.
139. Picard, R.W., Affective computing, 2000, M.I.T Media Laboratory: Cambridge, MA.
140. Dahlback, N., A. Jonsson, and L. Ahrenberg. Wizard of Oz studies: why and how. in Proceedings of the 1st international conference on Intelligent user interfaces. 1993, Orlando, Florida, United States: ACM.
141. Baum, L.F. and W.W. Denslow, The Wonderful Wizard of Oz The Oz Books. 1900: George M. Hill Company.
142. Fleming, V., The Wizard of Oz, 1939, Metro-Goldwyn-Mayer: USA. p. 101min.
143. Schlögl, S., A. Schneider, S. Luz, and G. Doherty. Supporting the wizard: interface improvements in Wizard of Oz studies. in Proceedings of the 25th BCS Conference on Human-Computer Interaction. Year: British Computer Society.
144. Molin, L. Wizard-of-Oz prototyping for co-operative interaction design of graphical user interfaces. in Proceedings of the third Nordic conference on Human-computer interaction. 2004, Tampere, Finland: ACM.
145. Shafer, S., J. Krumm, B. Brumitt, B. Meyers, M. Czerwinski, and D. Robbins. The new EasyLiving Project at Microsoft Research. in Joint DARPA/NIST Smart Spaces Workshop. Year, Gathersburg, Maryland.
146. Kindberg, T. and J. Barton, A Web-based nomadic computing system. Computer Networks, 2001. 35(4): p. 443-456.
147. Brumitt, B., B. Meyers, J. Krumm, A. Kern, and S. Shafer, EasyLiving: Technologies for Intelligent Environments, P. Thomas and H.-W. Gellersen, Editors. 2000, Springer Berlin / Heidelberg. p. 97-119.
148. Debaty, P. and D. Caswell, Uniform Web presence architecture for people, places, and things. 2001, New York, NY, ETATS-UNIS: Institute of Electrical and Electronics Engineers. 63.
149. Spasojevic, M. and T. Kindberg, Evaluating the CoolTown User Experience, Internet and Mobile Systems Laboratory Hewlett-Packard Laboratories: Palo Alto.
150. Cruz-Neira, C., D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart, The CAVE: audio visual experience automatic virtual environment. Commun. ACM, 1992. 35(6): p. 64-72.

151. Chung, J.C., M.R. Harris, F.P. Brooks, H. Fuchs, M.T. Kelley, J. Hughes, M. Ouh-young, C. Cheung, R.L. Holloway, and M. Pique. Exploring Virtual Worlds with Head-Mounted Displays. in Non-Holographic True 3-Dimensional Display Technolgies, SPIE Proceedings. Year: Los Angeles.
152. Bowman, D.A. and R.P. McMahan, Virtual Reality: How Much Immersion Is Enough? Computer, 2007. 40(7): p. 36-43.
153. Sutcliffe, A., B. Gault, and J.-E. Shin, Presence, memory and interaction in virtual environments. International Journal of Human-Computer Studies, 2005. 62(3): p. 307-327.
154. Johansson, P. and A. Ynnerman, Immersive Visual Interfaces--Assessing Usability by the Effects of Learning/Results from an Empirical Study. Journal of Computing and Information Science in Engineering, 2004. 4(2): p. 124-131.
155. Kasik, D.J., J.J. Troy, S.R. Amorosi, M.O. Murray, and S.N. Swamy, Evaluating graphics displays for complex 3D models. Computer Graphics and Applications, IEEE, 2002. 22(3): p. 56-64.
156. Swindells, C., B.A. Po, I. Hajshirmohammadi, B. Corrie, J.C. Dill, B.D. Fisher, and K.S. Booth. Comparing CAVE, wall, and desktop displays for navigation and wayfinding in complex 3D models. in Computer Graphics International, 2004. Proceedings. Year.
157. Mania, K., D. Wooldridge, M. Coxon, and A. Robinson, The Effect of Visual and Interaction Fidelity on Spatial Cognition in Immersive Virtual Environments. IEEE Transactions on Visualization and Computer Graphics, 2006. 12(3): p. 396-404.
158. Laurila, J.K., D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen. The mobile data challenge: Big data for mobile computing research. in Proceedings of the Workshop on the Nokia Mobile Data Challenge, in Conjunction with the 10th International Conference on Pervasive Computing. Year.
159. Silverman, D., Doing Qualitative Research; A Practical Handbook. 2 ed. 2000: SAGE.
160. Apple Inc. AirPlay. 2012 [cited 2012 19/09]; Available from: http://www.apple.com/airplay/.
161. Apple Inc. Introduction to Cocoa Bindings Programming Topics. Cocoa Bindings Programming Topics 2010 [cited 2013 30/04]; Available from: https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaBindings/CocoaBindings.html.
162. Apple Inc Interface Builder User Guide. Mac OS X, 2010.
163. Apple Inc Cocoa. Mac OS X, 2011.
164. Yamaha. Digital Mixers. 2012 [cited 2012 28/08]; Available from: http://uk.yamaha.com/en/products/proaudio/mixers/digital-mixers/01v96vcm/?mode=series.
165. OpenGL. 2012 [cited 2012 28/08]; Available from: http://www.opengl.org/.
166. Apple Inc. Graphics and Animation. OS X 2013 [cited 2013 30/04]; Available from: https://developer.apple.com/technologies/mac/graphics-and-animation.html.
167. Hada, N. and T. Ruark, Export Layers To Files, 2007, Adobe Systems.

168. Jorgensen, A.P., M. Collard, and C. Koch. Prototyping iPhone apps: realistic experiences on the device. in Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries. 2010, Reykjavik, Iceland: ACM.
169. Jackson, B.J.C., P.L. Watten, P.F. Newbury, and P.F. Lister. Rapid Authoring for VR-Based Simulations of Pervasive Computing Applications. in 1st International Workshop on Methods & Tools for Designing VR Applications (MeTo-VR) In conjunction with 11th International Conference on Virtual Systems and Multimedia. Year, Flanders Expo, Ghent, Belgium.
170. Axure. Axure. 2012 [cited 2012 30/09]; Available from: http://www.axure.com/.